Style Definition: Comment Text

Convolutional Neural Network and Long Short-Term Memory Models for Ice-Jam ~~Prediction~~Predictions

Fatemehalsadat Madaeni[1], Karem Chokmani[1], Rachid Lhissou[1], Saied Homayouni[1], Yves Gauthier[1], and Simon Tolszczuk-Leclerc[2]

[1]INRS-ETE, Université du Québec, Québec City, G1K 9A9, Canada
[2]EMGeo Operations, Natural Resources Canada, Ottawa, K1S 5K2, Canada
Correspondence to: Fatemehalsadat Madaeni (Fatemehalsadat.Madaeni@ete.inrs.ca)

Abstract. In cold regions, ice~~-jam events~~ jams frequently result in severe flooding due to a rapid rise in water levels upstream of the jam. ~~These~~Sudden floods resulting from ice jams threaten human safety, and cause damage to properties and ~~infrastructures as the floods resulting from ice-jams are sudden~~infrastructure. Hence, ice-jam prediction tools can give an early warning to increase response time and minimize the possible ~~corresponding~~ damages. However, ice-jam prediction has always been a ~~challenging problem~~challenge as there is no analytical method available for this purpose. Nonetheless, ice jams form when some hydro-meteorological conditions happen, a few hours to a few days before the event. Ice-jam prediction ~~problem~~ can be ~~considered~~addressed as a binary multivariate time-series classification. Deep learning techniques have been widely used for time-series classification in many fields such as finance, engineering, weather forecasting, and medicine. In this research, we successfully applied ~~Convolutional Neural Network~~ convolutional neural networks (CNN), ~~Long Short-Term Memory~~long short-term memory (LSTM), and combined ~~Convolutional-Long Short-Term Memory~~convolutional-long short-term memory (CNN-LSTM) networks ~~for~~ to predict the formation of ice~~-jam prediction for~~ jams in 150 rivers in the Province of Quebec (Canada). We also employed machine learning methods including support vector machine (SVM), k-nearest neighbors classifier (KNN), decision tree, and multilayer perceptron (MLP) for this purpose. The hydro-meteorological variables (e.g., temperature, precipitation, and snow depth) along with the corresponding jam or no-jam events are used as ~~the~~model inputs~~ to the models. We hold out 10%~~. Ten percent of the data was excluded from the model and set aside for testing~~. And we applied~~ , and 100 ~~re-shuffling~~reshuffling and splitting iterations ~~with~~ were applied to 80~~-~~% of the remaining data for training and 20% for validation. The developed deep learning models achieved improvements in performance in comparison to the developed machine learning models. The results show that the CNN-LSTM model yields the best results in the validation and testing with F1 scores of 0.82 and 0.92, respectively. This demonstrates that CNN and LSTM models are complementary, and a combination of ~~them~~both further improves classification.

Formatted: English (United Kingdom)

1 Introduction

Predicting ice~~-jam events~~jams gives an early warning of possible flooding events, but there is no analytical solution to predict these events due to the complex interactions between ~~involved~~the hydro-meteorological variables (e.g., temperature, precipitation, snow depth, and solar radiation~~).~~) involved. To date, a small number of empirical and statistical prediction methods such as threshold methods, multi-regression models, logistic regression models, and discriminant function analysis have been developed for ice jams (Barnes-Svarney and Montz, 1985; Mahabir et al., 2006; Massie et al., 2002; White, 2003; White and Daly, 2002~~, January~~; Zhao et al., 2012). However, these methods are site-specific ~~with a~~and have high ~~rate~~rates of false-positive errors (White, 2003). The numerical models developed for ice-jam prediction (e.g., ICEJAM (Flato and Gerard, 1986, cf.; Carson et al., 2011), RIVJAM (Beltaos, 1993), HEC-RAS (Brunner, 2002), ICESIM (Carson et al., 2001 and 2003), and RIVICE (Lindenschmidt, 2017)) ~~show~~have several limitations ~~in predicting ice-jam occurrence. This is because~~. More particularly, the mathematical formulations used in these models are complex ~~which~~and need many parameters ~~that~~, which are often unavailable as they are challenging to measure in ice conditions. ~~Hence, many~~The subsequent simplifications ~~corresponding~~necessary to ~~these parameters may degrade~~model application decrease model accuracy (Shouyu & Honglan, 2005). A detailed overview of the previous models for ice-jam prediction based on hydro-meteorological data ~~are~~is presented in Madaeni et al. (2020).

Prediction of ice-jam occurrence can be considered as a binary multivariate time-series classification (TSC) problem when the time series of various ~~hydro-meteorological~~hydrometeorological variables ~~(explained later)~~ can be used to classify jam or no jam events. Time-series classification (particularly multivariate) has been widely used in various fields, including biomedical engineering, clinical prediction, human activity recognition, weather forecasting, and finance. Multivariate time~~-~~ series provide more patterns and improve classification performance compared to univariate time~~-~~ series (Zheng et al., 2016). Time-series classification is one of the most challenging problems in data mining and machine learning.

Most existing TSC methods are feature-based, distance-based, or ensemble methods (Cui et al., 2016). Feature extraction is challenging due to the difficulty of handcrafting useful features to capture intrinsic characteristics from time-series data (Karim et al., ~~2019~~2019a; Zheng et al., 2014~~, June~~). Hence, distance-based methods work better in TSC (Zheng et al., 2014~~, June~~). Among the hundreds of methods developed for TSC, the leading classifier with the best performance was an ensemble nearest neighbor approach with dynamic time warping (DTW) ~~for many years~~ (Fawaz et al., ~~2019, July~~2019a; Karim et al., ~~2019~~2019a).

In the k-nearest neighbors (KNN) classifier, the ~~given~~ test instance is classified by ~~a~~the majority vote of its k ~~closest~~-nearest neighbors in the training ~~data~~dataset. The ~~KNN classifier needs all the data~~entire dataset is necessary to make a prediction based of KNN, which requires ~~high~~a lot of processing memory. Hence, it is computationally expensive and ~~could be slow if~~time-consuming when the database is large~~, and~~. It is also sensitive to irrelevant features and ~~the~~data scale ~~of the data~~. Furthermore, the number of neighbors ~~to include~~included in the algorithm should be carefully selected. The KNN classifier is very challenging to be used for multivariate TSC. The dynamic time warping approach is ~~a more~~ robust alternative for Euclidean distance (the most widely used time-series distance measure) to measure the similarity between two ~~given~~ time series by searching for an optimal alignment (minimum distance) between them (Zheng et al., 2016). However, the combined KNN with DTW is time-consuming and inefficient for long multivariate time~~-~~ series (Lin et al., 2012; Zheng et al., 2014~~, June). The traditional~~). Traditional classification and ~~classic~~ data mining algorithms developed for TSC have high computational complexity or low prediction accuracy. This is due to the size and inherent complexity of time series, seasonality, noise, and feature correlation (Lin et al., 2012).

There are some machine learning methods available for TSC such as KNN and support vector machine (SVM). However, the focus of this research is on the deep learning models that have greatly ~~impacted~~improved sequence classification ~~problems~~ and ~~they can also be used for~~ that perform well with multivariate TSC ~~with good performance.~~.

Deep learning methods ~~are able to consider two-dimensionality in~~work with 2-D multivariate time~~-~~ series and their deeper architecture could further improve ~~the~~ classification especially for complex problems~~, which is~~. This explains why ~~their results are~~deep learning methods generally have more accurate and robust results than other currently used methods (Wu et al., ~~2018a, April~~2018). However, ~~they are~~ their training is more time consuming and their interpretation is more difficult~~ to interpret~~.

Deep learning ~~is a type of~~involves neural networks that ~~uses~~use multiple layers where nonlinear transformation is used to extract

higher-level features from the input data. Although deep learning ~~in recent years showed~~has recently shown promising performance in various fields such as image and speech recognition, document classification, and natural language processing, only a few studies ~~employed~~were dedicated to using deep learning for TSC (Gu et al., 2018; Fawaz et al., ~~2019, July~~2019a). Various studies show that deep neural networks significantly outperform the ensemble nearest neighbor with DTW (Fawaz et al., ~~2019, July~~2019a). The main benefit of deep learning networks is automatic feature~~-~~ extraction, which reduces the need for expert knowledge ~~of the field~~ and removes engineering bias ~~in the~~during classification ~~task (Fawaz et al., 2019)~~ as the probabilistic decision (e.g., classification) is taken by the network~~.~~ (Fawaz et al., 2019b).

The most widely used deep neural networks for TSC are ~~Multi-Layer Perceptron~~multi-layer perceptron (MLP; i.e., fully connected deep neural networks), ~~Convolutional Neural Networks~~convolutional neural networks (CNNs), and ~~Long Short-Term Memory~~long short-term memory networks (LSTM). The application of CNNs for TSC has recently become ~~more and more~~increasingly popular, and different types of CNN are being developed with superior accuracy for this purpose (~~e.g.,~~ Cui et al., 2016). Zheng et al. (2014~~, June~~) and Zheng et al. (2016) introduce a ~~Multi-Channels Deep Convolutional Neural Network~~multi-channel deep convolutional neural network (MC-DCNN) for multivariate TSC, where each variable (i.e., univariate time series) is trained individually to extract features and finally concatenated using an MLP to perform classification (Fig. 1). ~~They~~The authors showed that their model achieves a state-of-the-art performance ~~both~~ in terms of efficiency and accuracy on a challenging dataset. The drawback of their model and similar architectures (e.g., Devineau et al., ~~2018, May~~2018a) is that they do not capture the correlation between variables as the feature extraction is carried out separately for each variable.

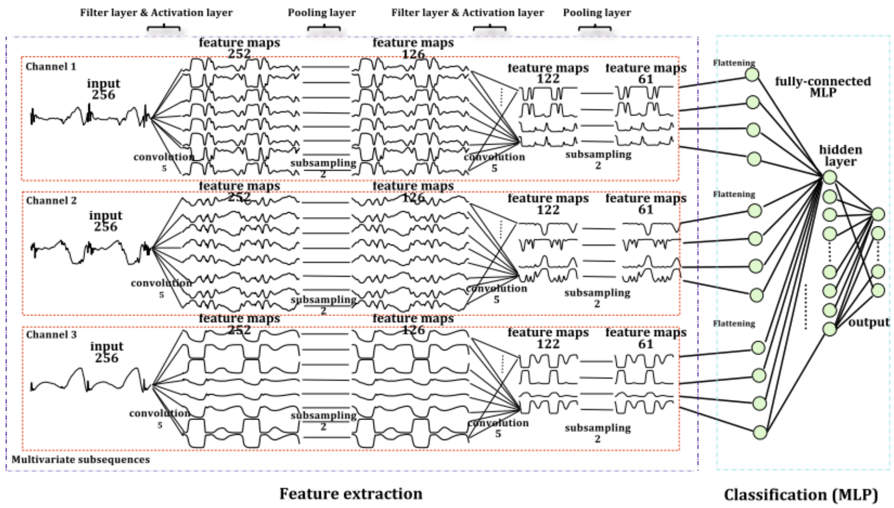
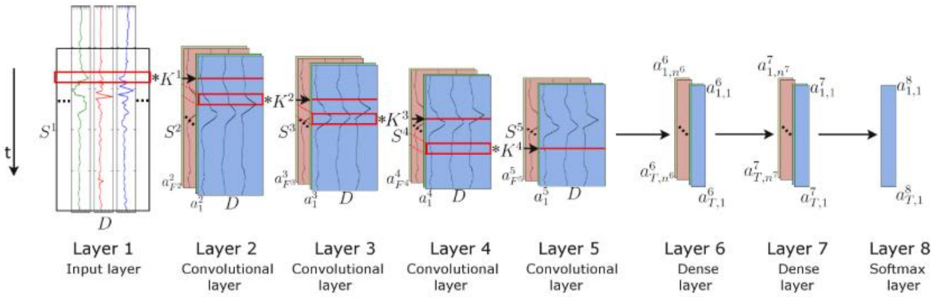


Figure 1. A ~~2 stages~~two-stage MC-DCNN architecture for activity classification. This architecture consists of a three ~~channels~~ channel input, two filter layers, two pooling layers, and two fully-connected layers (after Zheng et al., 2014~~, June~~).

Brunel et al. (2019) present CNNs adapted for TSC in cosmology using ~~1D~~1-D filters to extract features from each channel over time and a ~~1D~~ convolution in depth to capture the correlation between the channels. They compared the results from LSTMs with those from CNNs, ~~which shows~~and demonstrated that CNNs ~~give~~had better results ~~than LSTMs~~. Nevertheless, both deep learning approaches are very promising.

The combination of CNNs and LSTM units has already yielded ~~state-of-the-art~~promising results in problems requiring ~~classification of~~ temporal information classification, such as human activity recognition (Li et al., 2017; Mutegeki and Han, 2020~~, February~~), text classification (Luan and Lin, 2019; ~~March,~~ She and Zhang, 2018~~, December;~~ ; Umer et al., 2020), video classification (Lu et al., 2018 and Wu et al., 2015~~, October~~), sentiment analysis (Ombabi et al., 2020; Sosa, 2017; Wang et al., 2016~~, August~~; Wang et al., 2019), typhoon formation forecasting (Chen et al.,2019), and arrhythmia diagnosis (Oh et al., 2018). In this architecture, convolutional operations capture features and LSTMs capture time dependencies on the extracted features. Ordóñez and Roggen (2016) propose a deep convolutional LSTM model (DeepConvLSTM) for activity recognition (Fig. 2). Their results are compared to the results from standard feedforward units showing that DeepConvLSTM reaches a higher F1 score and better decision boundaries for classification. Furthermore, they noticed that the LSTM model ~~gives~~is also promising ~~results~~ with relatively small datasets. Furthermore, LSTMs ~~present a~~perform better ~~performance in capturing~~with longer temporal dynamics, whereas the convolution filters can only capture the temporal dependencies dynamics within the length of the filter.



Figure~~ ~~2. ~~The architecture~~Architecture of the DeepConvLSTM framework used for activity recognition (after Ordóñez and Roggen, 2016).

~~This~~The project presented in this paper is ~~a~~ part of a greater project called DAVE, which aims ~~to develop~~at developing a tool ~~to provide~~for regional ice jam watches and warnings, based on the integration of three aspects: ~~the~~ current ice cover conditions~~ of the ice cover~~; hydro-meteorological patterns associated with breakup ice jams; and channel predisposition to ice-jam formation. The outputs ~~of the previous tasks~~ will be used to develop an ice~~-~~ jam monitoring and warning module ~~and~~that will transfer the knowledge ~~gained~~ to the end~~-~~ users ~~to better manage the risk of~~managing ice ~~jams~~jam consequences.

The objective of this research is to develop deep learning models to predict breakup ice-jam events to be used as an early warning system of possible flooding. While most TSC research in deep learning is performed on ~~1D~~1-D channels (Hatami et al., 2018~~, April), we propose~~), our approach consists of using deep learning frameworks for multivariate TSC~~ for~~, applied to ice-jam prediction. Through our comprehensive literature review, we noticed that CNN (e.g., Brunel et al., 2019; Cui et al., 2016; Devineau et al., ~~2018, June~~2018b; Kashiparekh, 2019~~, July~~; Nosratabadi et al., 2020; Yan et al., 2020; Yang et al., 2015~~, June~~; Yi et al., 2017; Zheng et al., 2016), LSTM (e.g., Fischer and Krauss, 2018; Lipton et al., 2015; Nosratabadi et al., 2020; Torres et al., 2021), and a combined CNN-LSTM (e.g., Karim et al., ~~2o17~~2017; Livieris et al., 2020; Ordóñez and Roggen, 2016; Sainath et al., 2015~~, April~~; Xingjian et al., 2015) have been widely used ~~-~~for TSC. ~~There are numerous~~Numerous applications of CNN, LSTM, and their hybrid versions ~~applied~~are currently used in the field of hydrology (Althoff et al., 2021; Apaydin et al., 2020; Barzegar et al., 2021, 2020; Kratzert et al., 2018; Wunsch et al., 2020; Zhang et al., 2018). Although deep learning methods seem ~~to be~~ promising to address the requirements of ice-jam predictions, none of these methods yet have been explored for ice~~-~~ jam prediction.

~~Hence, we~~Although machine learning methods have been widely used in time series forecasting of hydro-meteorological data, they have been used less frequently in the prediction of ice jams (Graf et al., 2022). Semenova et al. (2020) used KNN to predict ice jams using hydro-meteorological variables such as precipitation, snow depth, water level, water discharge, and temperature. They developed their model with data collected from the confluence of Sukhona River and Yug River in Russia between 1960 and 2016 and achieved accuracy of 82%. Sarafanov et al. (2021) presented an ensemble-based model of machine learning methods and a physical snowmelt-runoff model to account for the advantages of physical models (interpretability) and machine learning models (low forecasting error).

Their hybrid models proposed an automated approach for short-term flood forecasting in Lena River, Poland, using hydro-meteorological variables (e.g., maximum water level, mean daily water and air temperature, mean daily water discharge, relative humidity, snow depth, and ice thickness). They applied an automated machine learning approach based on the evolutionary algorithm to automatically identify machine learning models, tune hyperparameters, and combine stand-alone models into ensembles. Their model was validated on ten hydro gauges for two years, showing that the hybrid model is much more efficient than stand-alone models with a Nash–Sutcliffe efficiency coefficient of 0.8. Graf et al. (2022) developed an MLP and extreme gradient boosting model to predict ice jams with data from 1983 to 2013, in Warta River, Poland. They employed water and air temperatures, river flow, and water level as inputs to their models, showing that both machine learning methods provide promising results. In Canada, De Coste et al. (2021) developed a hybrid model including a number of machine learning models (e.g., KNN, SVM, random forest, and gradient boosting) for St. John River (New Brunswick). The most successful ensemble model combining 6 different member models was produced with a prediction accuracy of 86% over 11 years of record.

We developed three deep learning models; a CNN, an LSTM, and a combined CNN-LSTM for ice-jam predictions, and compared the results. The previous studies show that these models ~~show good capabilities in capturing~~successfully capture features~~ and~~, the correlation between features (through convolution units) and time dependencies (through memory units) ~~that will be later~~which are subsequently used for TSC. The combined CNN-LSTM can reduce errors by compensating for the internal weaknesses of each model. In the CNN-LSTM model, CNNs capture features, then the LSTMs ~~give the~~identifies time dependencies on the captured features.

Furthermore, we also developed some machine learning methods as simpler methods for ice-jam prediction. And their results are compared with ~~results~~those obtained from the~~ developed~~ deep learning models.

2 Materials and Methods

2.1 Data and study area

It is known that specific hydro-meteorological conditions lead to ice-jam occurrence (Turcotte and Morse, 2015~~, August~~; and White, 2003). For instance, breakup ice jams occur when a period of intense cold is followed by a rapid peak discharge resulting from spring rainfall and snowmelt runoff (Massie et al., 2002). ~~The period of intense cold can be represented by the changes in~~ Accumulated ~~Freezing Degree Days~~freezing degree days (AFDD~~). The sudden~~) can be used as a proxy for intense cold periods. Sudden spring runoff increase, however, is not often available at the jam location and can be represented by liquid precipitation and snow depth ~~some~~a few days ~~before~~prior to ice-jam occurrence (Zhao et al., 2012). Prowse and Bonsal (2004) and Prowse et al. (2007) ~~evaluate~~assessed various hydroclimatic explanations for river ice freeze-up and breakup, concluding that shortwave radiation is the most critical factor influencing the mechanical strength of ice and consequently the possibility of breakup ice jams to occur. Turcotte and Morse (2015~~, August~~) explain that ~~Accumulated Thawing Degree Day~~accumulated thawing degree day (ATDD), an indicator of warming periods, partially covers the effect of shortwave radiation. ~~-~~In~~ the~~ previous studies ~~of~~addressing ice-jam and breakup predictions, discharge and changes in discharge, water level and changes in water level, AFDD, ATDD, precipitation, solar radiation, heat budget, and snowmelt or snowpack are the most ~~readily~~frequently used variables (Madaeni et al., 2020).

The inputs ~~we~~ used in this study are historical ice-jam or no ice-jam occurrence (Fig. 3) as well as hydro-meteorological variables ~~including~~from 150 rivers in Quebec, namely liquid precipitation (mm), ~~min~~minimum and ~~max~~maximum temperature (°C), AFDD (from August ~~1st~~1 of each year; °C), ATDD (from January ~~1st~~1 of each year; °C), snow depth (cm) and net radiation (W m^{-2}~~) in 150 rivers in Quebec.~~). The net solar radiation, which represents the total energy available to influence the climate, is calculated as the difference between incoming and outgoing energy. If the median temperature is greater than 1, ~~the~~ precipitation is considered to be liquid ~~precipitation.~~. The statistics of hydro-meteorological data used in the models are presented in Table 1. The source, time period, and spatial resolution of the input variables are shown in Table 2.

Table ~~1~~1. Statistics of hydro-meteorological variables used in the models.

Statistic	Liquid Precipitation (mm)	Minimum temperature (°C)	Maximum temperature (°C)	Net radiation (W m-2)	ATDD (°C)	AFDD (°C)	Snowdepth (cm)
Minimum	0.00	-40.00	-25.97	-67.77	0.00	-2109.33	0.00
Maximum	50.87	12.05	27.48	222.69	280.82	-35.41	121.86
Mean	1.04	-9.41	0.98	59.75	8.83	-898.48	15.99
Median	0.00	-7.73	1.68	59.41	1.27	-890.74	11.50

Table 2. Source, duration, and spatial resolution of hydro-meteorological data used in the models.

Data	Source	Duration	Spatial resolution
Min and Max temperature*	Daily Surface Weather Data (Daymet; Thornton et al., 2020)	1979–2019	1 km
Liquid precipitation	Canadian Precipitation Analysis (CaPA; Mahfouf et al., 2007)	2002–2019	10–15 km
Liquid precipitation	North American Regional Reanalysis (NARR; Mesinger et al., 2006)	1979–2001	30 km
Infrared radiation emitted by the atmosphere	North American Regional Reanalysis (NARR)	1979–2019	30 km
Infrared radiation emitted from the surface	North American Regional Reanalysis (NARR)	1979–2019	30 km
Snow depth	North American Regional Reanalysis (NARR)	1979–2019	30 km

* The average was used to derive the AFDD and the ATDD.

Ice-jam database ~~is~~was provided by the ~~Quebec Ministry of Public Security (~~provincial public safety department (Ministère de la sécurité publique du Québec; MSPQ; Données Québec, 2021) for 150 rivers ~~in Quebec~~, mainly in the St. Lawrence ~~basin~~River Basin. The database comes from ~~the~~ digital or paper ~~event reports~~events reported by local authorities under the jurisdiction of the MSPQ from 1985 to 2014. Moreover, ~~some~~ other data ~~of~~used to build this database ~~are~~were provided by ~~the~~ field observations ~~from~~collected by the Vigilance ~~/~~ /Flood application from 2013 to 2019. It contains 995 recorded jam events that are not validated and contain many inaccuracies, mainly in the toponymy of the rivers, location, dating, and ~~the~~jam event redundancy ~~of jam events~~.

The names of the watercourse of several recorded jams are not given ~~or completely,~~ wrong or ~~affected by a typo or an abbreviation~~misspelled. The toponymy of the rivers was corrected using the National Hydrographic Network (NHN; National Hydrographic Network - Natural Resources Canada (NRCan)), the ~~Geobase~~GeoBase of the Quebec ~~hydrographic network~~Hydrographic Network (National Hydro Network - NHN - GeoBase Series - Natural Resources

Formatted: Normal, Justified, Line spacing: 1.5 lines, Don't keep with next, Pattern: Clear (White)

Formatted: Font: 9 pt, Bold, Font color: Text 1, English (United States)

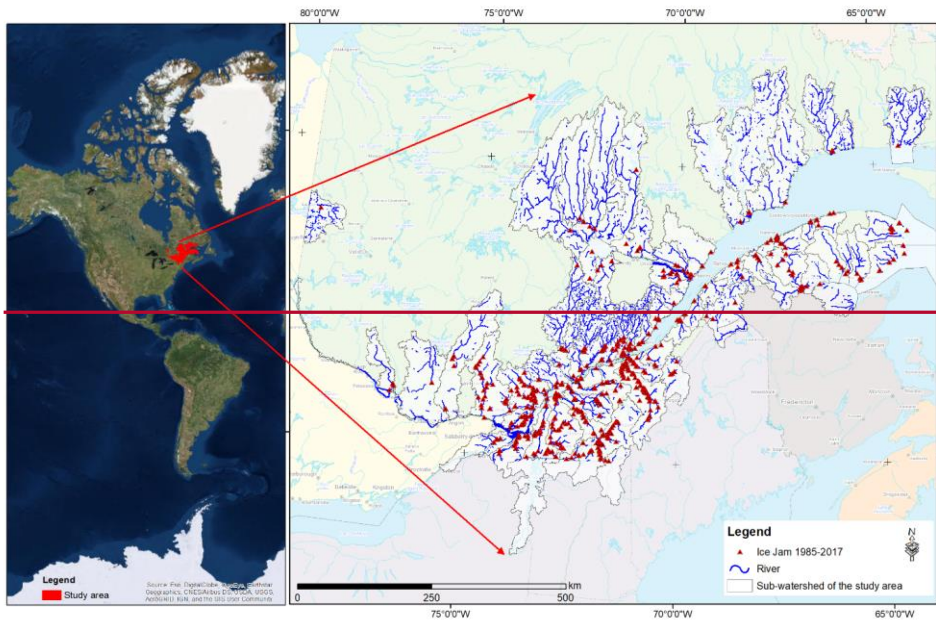
Formatted: Font: 10 pt, English (Canada)

Formatted: Justified, Line spacing: 1.5 lines, Pattern: Clear (White)

Canada), and the Toporama Web map service (The Atlas of Canada - Toporama - Natural Resources Canada) of the Sector of Earth Sciences.

~~Several~~Other manual corrections had to be carried out on ice jam data. For example, ice jams ~~are~~location is sometimes placed on the ~~banks~~riverbanks at a small distance (less than 20~~-~~ m) from the river polygon~~ of the river.~~. In this case, the location of the ice jam is manually moved inside the river polygon. In other cases, the ice-jam point is ~~posed~~placed further on the flooded shore at a distance between 20 m and 200~~ m.~~ m from the true ice jam. This ~~has been~~was corrected based on images with very high spatial resolution, based on the sinuosity and the narrowing of the river, the history of ice jams at the site~~ in question~~, and ~~the~~ press archives. In addition, some ice jams were placed too far from the ~~mentioned~~ river due to wrong~~ recorded~~ coordinates in the database. A single-digit correction in longitude or latitude returned the jam to its exact location. There are certain cases where the date of jam formation is verified by searching ~~the~~ press archives, notably when the date of formation is missing or several jams with the same dates and close locations in a section of a river are present.

The ice jam database contains many duplicates. This redundancy can be ~~due to~~ explained by the merging of two ~~data sources, the~~ databases, a double entry during ice~~-~~ jam monitoring, or ~~recording~~ numerous recordings for an ice jam that lasted for several days. ~~The~~To remediate this, the duplicates ~~are~~were removed from the database. The corrected ice-jam database contains 850 jams for 150 rivers, mainly in southern Quebec (Fig. 3). ~~The ice~~Ice jams formed in November and December (freeze-up jams) are removed ~~to only include~~from the model since the processes involved are different from breakup ice jams (included from January ~~15th) in the modelling as these two types of jams are formed due to different processes.~~15). The final breakup ice-jam database~~ that~~ used in this study includes 504 jam events.



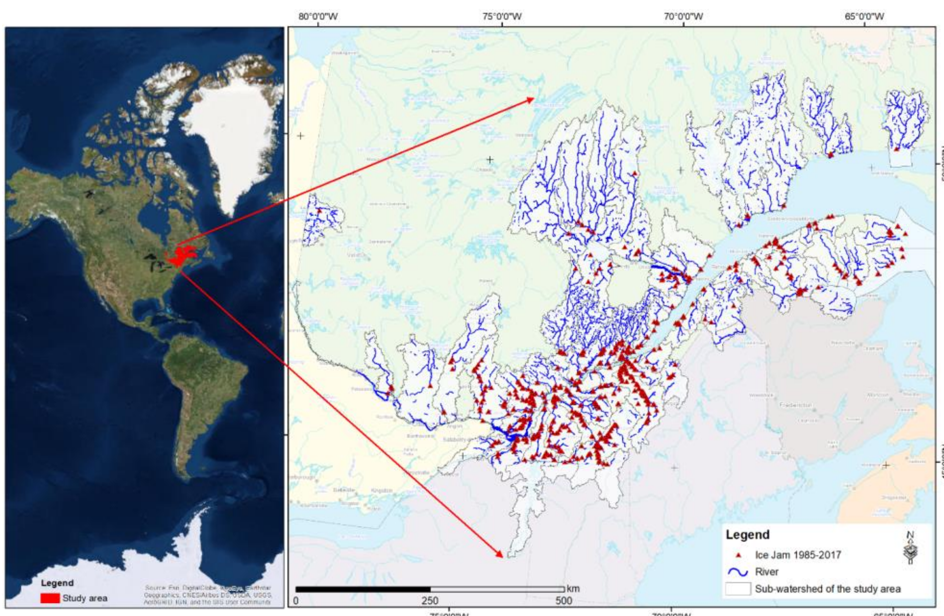


Figure 3. Study area and historic ice-jam locations recorded in Quebec from 1985- to 2017.

~~Table 1. Statistics of hydro-meteorological variables used in the models.~~

~~Statistics~~	~~Liquid P (mm)~~	~~Tmin (°C)~~	~~Tmax (°C)~~	~~Net radiation (W m-2)~~	~~ATDD (°C)~~	~~AFDD (°C)~~	~~Snowdepth (cm)~~
~~min~~	~~0.00~~	~~-40.00~~	~~-25.97~~	~~-67.77~~	~~0.00~~	~~-2109.33~~	~~0.00~~
~~max~~	~~50.87~~	~~12.05~~	~~27.48~~	~~222.69~~	~~280.82~~	~~-35.41~~	~~121.86~~
~~average~~	~~1.04~~	~~-9.41~~	~~0.98~~	~~59.75~~	~~8.83~~	~~-898.48~~	~~15.99~~
~~median~~	~~0.00~~	~~-7.73~~	~~1.68~~	~~59.41~~	~~1.27~~	~~-890.74~~	~~11.50~~

~~Table 2. Source, duration, and spatial resolution of hydro-meteorological data used in the models.~~

~~Data~~	~~Source~~	~~Duration~~	~~Spatial resolution~~
~~Min and Max temperature*~~	~~Daily Surface Weather Data (Daymet; Thornton et al., 2020)~~	~~1979-2019~~	~~1 km~~
~~Liquid precipitation~~	~~Canadian Precipitation Analysis (CaPA; Mahfouf et al., 2007)~~	~~2002-2019~~	~~10-15km~~
~~Liquid precipitation~~	~~North American Regional Reanalysis (NARR; Mesinger et al., 2006)~~	~~1979-2001~~	~~30 km~~
~~Infrared radiation emitted by the atmosphere~~	~~North American Regional Reanalysis (NARR)~~	~~1979-2019~~	~~30 km~~
~~Infrared radiation emitted from the surface~~	~~North American Regional Reanalysis (NARR)~~	~~1979-2019~~	~~30 km~~
~~Snow depth~~	~~North American Regional Reanalysis (NARR)~~	~~1979-2019~~	~~30 km~~

~~* The average was used to derive the AFDD and the ATDD.~~

2.2 Machine learning models for TSC

The most common machine learning techniques ~~that have been~~ used for TSC are SVM (Rodríguez and Alonso, 2004; Xing and Keogh, 2010), KNN (Li et al., 2013; Xing and Keogh, 2010), decision tree (DT; Brunello et al., 2019; Jović et al., 2012~~, August~~), and multilayer perceptron (MLP; del Campo et al., 2021; Nanopoulos et al., 2001). ~~For more information about these machine learning models refer to the mentioned literature above. We do not explain these~~These models and their applications in TSC~~, as they~~ are ~~not~~beyond the ~~focus~~scope of this study and will not be further addressed.

We developed the mentioned machine learning methods and compared their results with ~~the results~~those of deep learning models. After some trials and errors, the parameters that are changed from the default values for each machine learning model are as follows. We developed an SVM with a polynomial kernel with a degree of 5 that can distinguish curved or nonlinear input space. The KNN is used with 3 neighbors used for classification. The decision tree model is applied with all the default values. The shallow MLP is used with ~~'lbfgs'~~"lbfgs" solver (which can converge faster and perform better for small datasets), alpha of ~~1e~~1 e-5, and 3 layers with 7 neurons in each layer.

2.3 Deep learning models for TSC

The most common and popular deep neural networks for TSC are MLPs, CNNs, and LSTMs (Brownlee, ~~2018~~2018b; and Torres et al., 2021). ~~Despite their power, however~~Although it is very powerful approach, MLP ~~has limitations~~networks are limited by the fact that each input (i.e., time-series element) and output are treated independently, which means that the temporal or space information is lost (Lipton et al., 2015). Hence, an MLP needs some temporal information in the input data to model sequential data, such as time series (Ordóñez and Roggen, 2016).

Formatted: Font: 9 pt, Bold, Font color: Text 1, English (United States)

Formatted: Normal, Justified, Line spacing: 1.5 lines, Don't keep with next, Pattern: Clear (White)

Formatted: Font: 10 pt, English (Canada)

Formatted: Justified, Line spacing: 1.5 lines, Pattern: Clear (White)

Formatted: Caption, Pattern: Clear

Formatted: Font color: Text 1

Formatted: Normal, Left, Space Before: 0 pt, After: 0 pt, Pattern: Clear (White)

In this regard, ~~Recurrent Neural Networks~~recurrent neural networks (RNNs) are specifically adapted to sequence data through the direct connections between individual layers (Jozefowicz et al., 2015). Recurrent ~~Neural Networks~~neural networks perform the same repeating function with a straightforward structure, e.g., a single tanh (hyperbolic tangent) layer, for every input of data (xt), ~~while all~~and the inputs are related to each other ~~with~~within their hidden internal state, which allows it to learn the temporal dynamics of sequential data (Fig. 4).

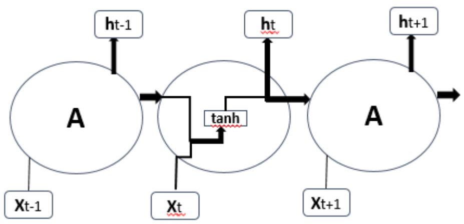


Figure 4. An RNN with a single tanh layer, where *A* is a chunk of the neural network, *x* is input data, and *h* is output data.

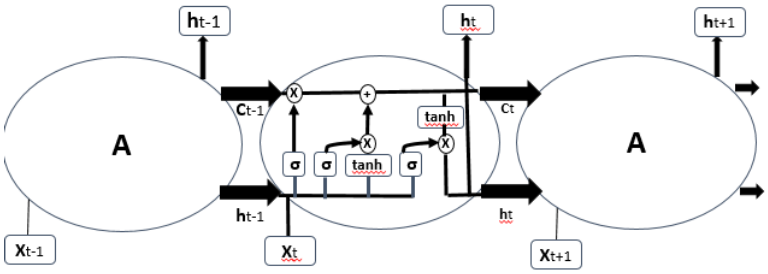
Recurrent ~~Neural Networks were~~ neural networks are rarely used in TSC due to their significant ~~problems. Recurrent Neural Networks mainly~~limitations: RNNs mostly predict ~~output~~outputs for each time-series element~~,~~; they are sensitive to the first examples seen, and it is~~ also~~ challenging to capture long-term dependencies due to vanishing gradients, exploding gradients, and their complex dynamics (Devineau et al., ~~2018, June~~2018b; Fawaz et al., ~~2019~~2019b).

Long short-term memory RNNs are developed to improve the performance of RNNs by integrating a memory component to model long-term dependencies in time-series problems (Brunel et al., 2019; Karim et al., ~~2019~~2019a). Long short-term memory networks do not have the problem of exploding gradients. The LSTMs have four interacting neural network layers in a very special way (Fig. 5). An LSTM has three sigmoid (σ) layers to control how much of each component should be let through by outputting numbers between zero and one. The input to an LSTM goes through three gates (“forget”, “input”, and “output gates”) that control the operation performed on each LSTM block (Ordóñez and Roggen, 2016). The first step is the “forget gate” layer that gets the output of the previous block (ht−1), the input for the current block (Xt), and the memory of the previous block (Ct-1) and gives a number between 0 and 1 for each number in the cell state (Ct−1; Olah, 2015). The second step is called the “input gate” with two parts, a sigmoid layer that decides which values to be updated and a tanh layer that creates new candidate values for the cell state. ~~These two~~The new and old memories ~~will~~are then be combined and control how much the new memory should influence the old memory. The last step (output gate) gives the output by applying a sigmoid layer deciding how much new cell memory goes to output, and ~~multiply~~multiplies it by tanh applied to the cell state (~~giving~~resulting in values between −1 and 1).

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic



Figure~~ ~~5. Structure of LSTM block with four interacting layers.

Recently, convolutional neural networks challenged the assumption that RNNs (e.g., LSTMs) have the best performance when working with sequences. ~~The CNNs show state-of-the-art performance in~~Convolutional NNs perform well when processing sequential data such as speech recognition and sentence classification, similar to TSC (Fawaz et al., ~~2019~~2019b).

~~The CNNs~~Convolutional NNs are the most widely used deep learning methods in TSC problems (Fawaz et al., ~~2019~~2019b). They learn spatial features from raw input time series using filters (Fawaz et al., ~~2019). The CNNs~~2019b). Convolutional NNs are robust and need a relatively small amount of training time ~~comparing with~~compared to RNNs or MLPs. They work best for extracting local information and reducing the complexity of the model.

A CNN is a kind of neural network with at least one convolutional (or filter) layer. A CNN usually involves several convolutional layers, activation functions, and pooling layers for feature extraction~~ following~~, followed by dense layers ~~(or MLP)~~used as ~~a classifier~~classifiers (Devineau et al., ~~2018, June~~2018b). The reason to use a sequence of filters is to learn various features from time series for TSC.~~ ~~ A convolutional layer consists of a set of learnable filters that compute dot products between local regions in the input and corresponding weights. With high-dimensional inputs, it is impractical to connect subsequent neurons to all the neurons ~~in~~of the previous layer. Therefore, each neuron in CNNs is connected to only a local region of the input~~, namely the ~~ receptive field~~, which equals ~~ whose size is equivalent to that of the filter ~~size ~~(Fig. 6). This feature reduces the number of parameters by limiting the number of connections between neurons in different layers. The input is first convolved with a learned filter, and then an element-wise nonlinear activation function is applied to the convolved results (Gu et al., 2018). The pooling layer performs a downsampling operation such as maximum or average, reducing the spatial dimension. One of the most powerful features of CNNs is called weight or parameter sharing, where all neurons share filters (weights) in a particular feature map (Fawaz et al., ~~2019)~~2019b). This allows to reduce the number of parameters.

Figure~~ ~~ 6. ~~A~~Structure of a convolution layer~~ structure~~ including two sets of filters.

Formatted: Font: 9 pt

2.4 Model libraries

In an Anaconda (Analytics, C., 2016) environment, Python is implemented to develop CNN, LSTM, and CNN-LSTM networks for TSC. To build and train networks, the networks are implemented in Theano (Bergstra et al., 2010~~, June~~) using the Lasagne library (Dieleman et al., 2015~~) library. The other~~). Other core libraries used for importing, preprocessing, training data, and visualization of results ~~are~~include Pandas (Reback et al., 2020), NumPy (Harris et al., 2020), Scikit-Learn (Pedregosa et al., 2011), and Matplotlib. PyLab (Hunter, J.~~-~~ D., 2007). The Spyder (Raybaut, 2009) package of Anaconda ~~is utilized~~can be used as an interface~~, or~~ otherwise, the command window can be used without any interface.

To develop machine learning models, Scikit-Learn machine learning libraries are used except for NumPy, Pandas, and Scikit-Learn preprocessing libraries.

2.5 Preprocessing

The data is comprised of variables with varying scales, and the machine~~-~~ learning algorithms can benefit from rescaling the variables to all have ~~the same~~one single scale. Scikit-learn (Pedregosa et al., 2011) is a free library for machine learning in Python that can be used to preprocess data. We examined Scikit-learn MinMaxScaler (scaling each variable between 0 and 1), Normalizer (scaling individual samples to the unit norm), and StandardScaler (transforming to zero mean and unit variance separately for each feature). The results show that MinMaxScaler (Eq. (1)) leads to the most accurate results. ~~The scaling of validation~~Validation data rescaling is ~~done with min~~carried out based on minimum and ~~max from~~maximum values of the train data.

$$X_{scaled} = \cancel{\frac{X - X.min}{X.max - X.min}}\frac{X - Xmin}{Xmax - Xmin} \quad (1)$$

For each jam or no jam event, ~~we used~~ the data from 15 days ~~of information before~~preceding the event was used to predict the event on the 16th day. ~~We generate a~~A balanced dataset with the same number of jam and no-jam events (1008 small sequences~~ totally),~~) was generated, preventing the model from becoming biased to jam or no-jam events. The hydro-meteorological data related to no-jam events ~~are~~were constructed by extracting data from the reaches of no-jam records. ~~To examine models' generalization, we hold out~~Model generalizations were assessed by extracting 10% of data for testing~~ and 80 % and 20 % of~~. With the remaining data, 80% was used for training and 20% for validation~~, respectively~~. We used ShuffleSplit subroutine from the Scikit-learn library, where the database was randomly sampled during each re-shuffling and splitting iteration to generate training and validation sets. We applied 100 re-shuffling and splitting iterations for training and validation. There are 726, 181, and 101 small sequences with the size of (16, 7), 16 days of data for the seven variables; for training, validation, and test, respectively.

2.6 Training

Training a deep neural network with an excellent generalization to new unseen inputs is challenging. As a benchmark, a CNN model with~~ the~~ parameters and layers similar to previous studies (e.g., Ordóñez and Roggen, 2016) is developed. The model shows underfitting or overfitting with various architectures and parameters. To overcome underfitting, deeper models and more nodes ~~in~~ can be added to each layer~~ are beneficial~~; however, overfitting is more challenging to ~~overcome. Ice~~resolve. The ice-jam dataset for Quebec contains 1008 balanced sequence instances (with a length of 16), which is ~~small for~~considered to be a small amount of data in the context of deep learning. ~~The deep~~Deep learning models ~~often~~ tend to overfit small datasets by memorizing inputs rather than training~~, as a~~. This is due to the fact that small ~~dataset~~datasets may not appropriately describe the relationship between input and output spaces.

2.6.1 Overcome overfitting

There are various ~~methods~~ways to ~~tackle~~resolve the problem of overfitting, including acquiring more data, data augmentation (e.g., cropping, rotating, and noise injection), dropout (Srivastava et al., 2014), early stopping, batch normalization (Ioffe and Szegedy, 2015~~, June~~), and regularization. Acquiring more data is not possible with ice-jam records. We added the Gaussian noise layer (from the Lasagne library), where the noise values are Gaussian-distributed with zero-mean and a standard deviation of 0.1 to the input. The noise layers applied to the CNN and LSTM models significantly overcome the overfitting problem through data augmentation. However, the performance of the CNN-LSTM model dramatically deteriorates~~, including~~ when a noise layer is added (Fig. 7). Adding a noise layer to other layers does not improve any of the developed models for ice-jam prediction.

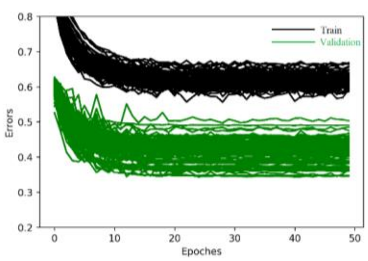
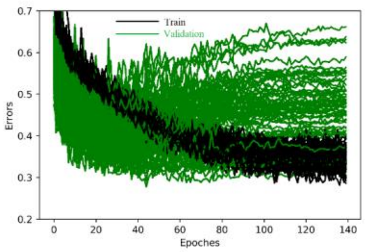


Figure 7. Train and validation errors over epochs for CNN-LSTM model with a noise layer.

Early stopping is another efficient method that halts the training procedure at a point where further training would decrease training loss, ~~while~~but validation loss starts to increase. Neural networks ~~solve an~~need a loss function to guide optimization problem ~~that requires a loss function to calculate the model error~~resolution. The loss function is similar to an objective function for process-based hydrological models. Among the developed models, only LSTM needs early stopping at 40 ~~epoch~~epochs (Fig. 8). More detailed explanations about the ~~other~~ methods that are used in this study to overcome overfitting (e.g., batch normalization~~,~~ and L2 regularization) can be found in the Appendix.



Figure~~ ~~ 8. Train and validation errors over epochs for an LSTM model showing overfitting after 40 epochs.

2.6.2 Model ~~Hyperparameters~~hyperparameters

Finding hyperparameter values ~~in deep learning~~ has been challenging due to the complex architecture of deep learning models and ~~a~~the large number of parameters (Garbin et al., 2020). ~~To find the~~The best model architecture~~, we study the~~ was identified by assessing model performance ~~of models~~ with different ~~layers~~layer and parameters such as the number of layers (noise, batch normalization, convolutional, pooling, LSTM, dropout, and dense~~ layers,~~) as well as different pooling sizes and strides, different batch sizes, various scaling of data (standardization and normalization), various filter sizes, number of units in LSTM and dense layers, the type of the activation functions, regularization and learning rates, weight decay and number of filters in convolutional layers. ~~We also applied various combinations of these layers and parameters. The hyperparameters~~Hyperparameters are optimized through manual trial and error searches as grid search experiments suffer from poor dimensional coverage in dimensions (Bergstra and Bengio, 2012~~)~~ ~~and~~). Another reason is that manual experiments are much easier to conduct and ~~more interpretable in~~interpret when investigating the effect of one hyperparameter of interest. The optimized hyperparameters are presented in Table~~ ~~ 3. The most important parameters of the models are explained below and ~~for more~~additional information ~~about other parameters readers are referred to~~is available in the Appendix.

Table~~ ~~ 3. Common ~~values~~ and selected values for different parameters of the models.

Parameter	Common values	Selected value	Source
Mini-batch size	16, 32, 64	16	Bengio (2012); Devineau et al. (2018b); Masters and Luschi (2018)
Number of convolution filters	32, 64, 128	128	Brownlee (2017); Maggiori et al. (2017)
Filter size	3, 5, 7	(5,1) and (5,3)	Devineau et al. (2018b); Maggiori et al. (2017)
Number of LSTM units	32, 64, 128	128	Brownlee (2017); Karim et al. (2019b); Ordóñez and Roggen (2016)
Number of dense layer units	16, 32, 128, 256	32	Karim et al. (2019a); Livieris et al. (2020); Fawaz et al. (2019b)
Momentum in SGD	0.5, 0.99, 0.9	0.9	Brownlee (2018a)

Inserted Cells

Formatted Table

Formatted: Left

Formatted: Left

Formatted: Left

Formatted: Left

Formatted: Left

Formatted: Left

Formatted Table

2.6.2.1 Number of layers

The depth of models is related to the sequence length (Devineau et al., ~~2018, May~~2018a), as deeper networks need more data to provide better generalization (Fawaz et al., ~~2019, July~~2019a). In ~~the~~ previous studies ~~of~~focused on CNNs, there ~~are~~were usually one, two, or three convolution stages (Zheng et al., 2014~~, June~~). We tried different numbers of CNN, LSTM, and dense layers and ~~selected~~the best combination was obtained with three CNN layers, two LSTIM layers, and two ~~such~~dense layers~~, respectively, as the~~. The sequence length in this study is small (16), and ~~we could not improve the~~ model performance was not improved by ~~merely adding more~~simply increasing depth.

2.6.2.2 Number and size of convolution filters

Data with more classes need more filters, and longer time series need longer filters to capture longer patterns and consequently to produce accurate results (Fawaz et al., ~~2019, July~~2019a). However, longer filters significantly increase the number of parameters and potential for overfitting small datasets, while a small filter size risks poor performance. ~~We finally selected~~In this study, two convolutional layers with 1-D filters of size (5, 1) and stride of (1, 1) to capture temporal variation for each variable separately. Furthermore, one convolutional layer with 2-D filters of size (5, 3) and stride of (1, 1) ~~is then~~was used to capture the correlation between variables via depth-wise convolution of input time~~-~~ series. A big stride might cause the model to miss valuable data used in predicting and smoothing out the noise in the time series. The layers in CNNs have a bias for each channel, sharing across all positions in each channel.

2.6.2.4 Adaptive learning rates

The adaptive learning rate decreases the learning rate and consequently weights over each epoch. We tried different base learning and decay rates for each model and found that the learning rate significantly impacts the ~~model~~model's performance. Finally, we chose a base learning rate of 0.1, 0.01, and 0.001 for LSTM, CNN, and CNN-LSTM, respectively. A decay rate of 0.8 was used for CNN and CNN-LSTM, ~~while~~and a rate of 0.95 for the LSTM model~~, this rate was 0.95.~~. Table~~-~~ 4 shows the adaptive learning rates for CNN, LSTM, and CNN-LSTM calculated using ~~Eq. (~~Equation 2~~)~~ for each epoch.

$$\text{adaptive learning rate} = base\ learning\ rate\ \times\ decay^{epoch} \qquad (2)$$

The experiments show that the learning rate is the most critical parameter influencing the model performance. A small learning rate can cause the loss function to get stuck in local minima, and a large learning rate can result in oscillations around global minima without reaching it.
Our CNN-LSTM model is deeper than the other two models, and deeper models are more prone to a vanishing gradient problem. To overcome the vanishing gradients, it is generally recommended ~~that~~to use lower learning rates, e.g., lower than ~~1e~~1 e-4~~, be used~~. Interestingly, we found that our CNN-LSTM model works better with lower learning rates than the other two models.

Table~~-~~ 4. The adaptive learning rate for 50 epochs.

	Learning rate

Epochs	CNN	CNN-LSTM	LSTM
1	0.008	8.00E-04	0.095
2	0.006	6.40E-04	0.09
3	0.005	5.12E-04	0.086
4	0.004	4.10E-04	0.081
.	.		.
.	.		.
40	1.30E-06	1.33E-07	0.013
.	.		-
50	1.40E-07	1.43E-08	-

Formatted: Font: 10 pt

2.6.5 Model evaluation

The network on the validation set is evaluated after each epoch during training to monitor the training progress. During validation, all non-deterministic layers are switched to deterministic. For instance, noise layers are disabled, and the update step of the parameters is not performed.

The classification accuracy cannot appropriately represent ~~the ~~model performance for unbalanced datasets, as the model can show a high accuracy by biasing towards the majority class in the dataset (Ordóñez and Roggen, 2016). While we built a balanced dataset (with the same number of jam and no jam events), randomly selecting test data~~ and~~, shuffling the inputs, and splitting data into train and validation sets can result in a slightly unbalanced dataset. In our case, the number of jams and no jams for train and validation and test sets is presented in Table 5. Therefore, the F1 score (Eq. (3)), which considers each class equally important, is used to measure the accuracy of binary classification. The F1 score, as a weighted average of the precision (Eq. (4)) and recall (Eq. (5)), ~~has the best~~ranges between 0 and ~~worst scores of ~~1, where 0 is the worse score and ~~0, respectively.~~1 is the best. In ~~Eqs. 7~~equations 4 and ~~8~~5, TP, FP, and FN are true positive, false positive, and false negative, respectively.

Table 5. The number of jam and no jam events used in ~~train~~the rain and validation and test datasets.

	Train and validation	Test
Jam	456	48
No jam	451	53

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (3)$$

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

$$Recall = \frac{TP}{TP+FN} \quad (5)$$

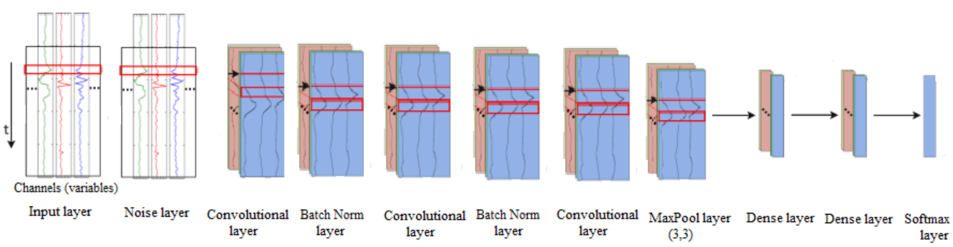
Although the model accuracy is usually used to examine the performance of deep learning models, the model size (i.e., number of parameters) provides a second metric, which represents required memory and calculations, to be compared among models with the same accuracy (Garbin et al., 2020).

After training the model, the well-trained network parameters are saved to a file and are later used ~~for testing the~~ to test network ~~generalization~~generalizations using ~~a~~the test dataset, ~~which~~composed of data that is not ~~seen~~used during training and validation.

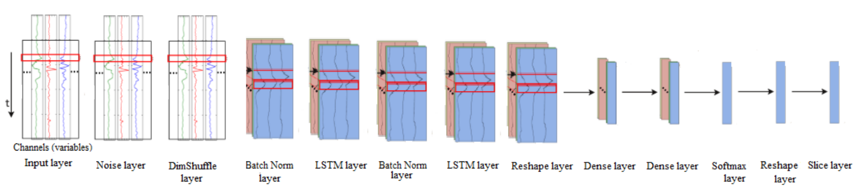
2.7 Architecture of models

The ~~architectures~~final architecture of CNN, LSTM, and CNN-LSTM models ~~that are finally selected~~ are presented in ~~Figs~~Figures. 9, 10, and 11, respectively. The layers, their output shapes, and their number of parameters are respectively presented in Tables~~ ~~ 6, 7, and 8 for CNN, LSTM, and CNN-LSTM models~~, respectively~~.

~~The CNN~~Convolutional NN models often include pooling layers to reduce data complexity and dimensionality. However, it is not always necessary ~~that~~for every convolutional layer ~~is~~to be followed by a pooling layer in the time-series domain (Ordóñez and Roggen, 2016). For instance, Fawaz et al. (~~2019, July~~2019a) do not apply any pooling layers in their TSC models~~ for TSC~~. We tried max-pooling layers after different convolutional layers in CNN and CNN-LSTM networks and found that a pooling layer following only the last convolutional layer improves the performance of both models. This can be due to subsampling the time series and using time series with a length of 16 that ~~reduces~~eliminates the need for ~~reducing~~decreased dimensionality.



Figure~~ ~~ 9. ~~The architecture~~Architecture of the CNN model for ice-jam prediction (adapted after Ordóñez and Roggen, 2016).



Figure~~ ~~ 10. ~~The architecture~~Architecture of the LSTM model for ice-jam prediction (adapted after Ordóñez and Roggen, 2016).

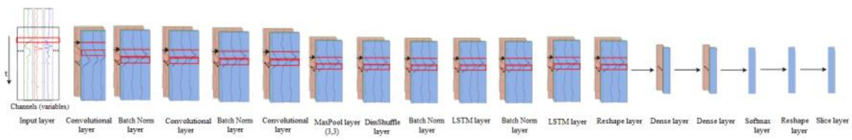


Figure 11. ~~The architecture~~Architecture of the CNN-LSTM model for ice-jam prediction (adapted after Ordóñez and Roggen, 2016).

Table 6. ~~The layers, their~~Layers, output shapes, and ~~their~~ number of parameters for the CNN model.

Layers	Output shape	Number of parameters
Input	(16, 1, 16, 7)	0
GaussianNoise	(16, 1, 16, 7)	0
Conv2D	(16, 128, 16, 7)	640
BatchNorm	(16, 128, 16, 7)	512
Nonlinearity	(16, 128, 16, 7)	0
Conv2D	(16, 128, 16, 7)	81920
BatchNorm	(16, 128, 16, 7)	512
Nonlinearity	(16, 128, 16, 7)	0
Conv2D	(16, 128, 16, 7)	245888
MaxPool2D	(16, 128, 5, 2)	0
Dense	(16, 32)	40992
Dense	(16, 32)	1056
Softmax	(16, 2)	66

Table 7. ~~The layers, their~~Layers, output shapes, and ~~their~~ number of parameters for the LSTM model.

Layers	Output shape	Number of parameters
Input	(16, 1, 16, 7)	0
GaussianNoise	(16, 1, 16, 7)	0
Dimshuffle	(16, 16, 1, 7)	0
BatchNorm	(16, 16, 1, 7)	64
LSTM	(16, 16, 128)	70272
BatchNorm	(16, 16, 128)	64
Nonlinearity	(16, 16, 128)	0
LSTM	(16, 16, 128)	132224
Reshape	(256, 128)	0
Dense	(256, 32)	4128
Dense	(256, 32)	1056
Softmax	(256, 2)	66
Reshape	(16, 16, 2)	0
Slice	(16, 2)	0

Formatted Table

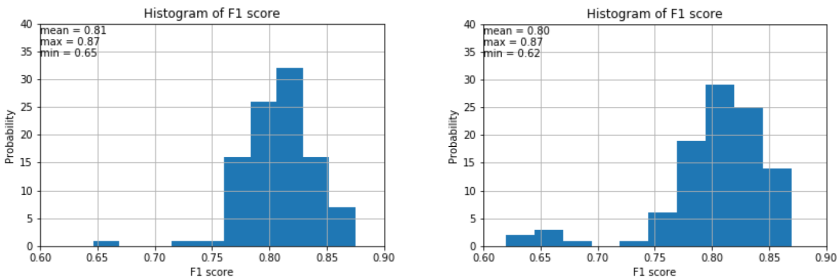
Table 8. ~~The layers, their~~Layers, output shapes, and ~~their~~ number of parameters for the CNN-LSTM model.

Layers	Output shape	Number of parameters
Input	(16, 1, 16, 7)	0
Conv2D	(16, 128, 16, 7)	640
BatchNorm	(16, 128, 16, 7)	512
Nonlinearity	(16, 128, 16, 7)	0
Conv2D	(16, 128, 16, 7)	81920
BatchNorm	(16, 128, 16, 7)	512
Nonlinearity	(16, 128, 16, 7)	0
Conv2D	(16, 128, 16, 7)	245888
MaxPool2D	(16, 128, 5, 2)	0
Dimshuffle	(16, 5, 128, 2)	0
BatchNorm	(16, 5, 128, 2)	20
LSTM	(16, 5, 128)	197760
BatchNorm	(16, 5, 128)	20
Nonlinearity	(16, 5, 128)	0
LSTM	(16, 5, 128)	132224
Reshape	(80, 128)	0
Dense	(80, 32)	4128
Dense	(80, 32)	1056
Softmax	(80, 2)	66
Reshape	(16, 5, 2)	0
Slice	(16, 2)	0

3 Results and Discussion

3.1 Weight initialization

Among all the ~~various types of~~ methods available ~~in Lasagne~~ for weight initialization~~,~~, in the Lasagne library, GLOROT uniform (i.e., Xavier~~;~~, Glorot and Bengio, 2010~~, March~~) and He initializations (He et al., 2015~~),~~) are the most popular initialization techniques~~, are used~~ to set the initial random weights in convolutional layers. The results reveal that in our case, these methods yield ~~almost the same~~comparable F1 scores. However, the histograms of F1 scores reveal that GLOROT uniform yields slightly better results (Fig. 12).



Figure~~-~~ 12. Histograms of F1 score for CNN using He (left) and GLOROT uniform (right) weight initialization with 100 random train-validation splits.

3.2 Model evaluation

3.2.1 Learning curves and F1 scores

Line plots of the loss (i.e., learning curves), which are loss over each epoch, are widely used to ~~examine the~~assess model performance ~~of models~~ in machine learning. Furthermore, line plots clearly indicate common learning problems, such as underfitting ~~or~~and overfitting. The learning curves for CNN, LSTM, and CNN-LSTM models are presented in Fig. 13. The LSTM model starts to overfit at epoch~~-~~ 40, so an early stopping is conducted. CNN-LSTM performs better than the other two models, as its training loss is the lowest and is lower than its validation loss. Histograms of F1 scores (Fig. 14 and Table 9) show that CNN-LSTM outperforms the other two models since it results in the highest average and the highest minimum F1-scores for validation (0.82 and 0.75, respectively). Figure~~-~~ 13 shows that the training error of the CNN model is lower than that of the LSTM~~, which means~~ model, indicating that ~~CNN~~it trained ~~better than LSTM model.~~more efficiently. However, it is not true for the validation error. The~~ reason that the~~ validation error is less than the training error in the LSTM model ~~can be the employment~~because of the regularization methods ~~as~~used. As LSTM models are often harder to regularize, agreeing with previous studies (e.g., Devineau et al., ~~2018, June~~2018b).

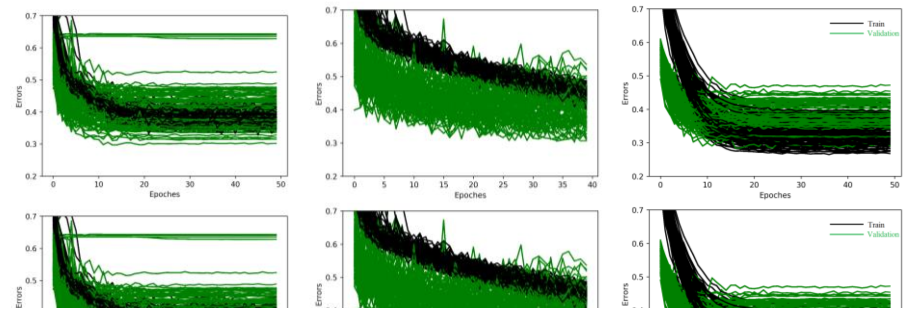
The LSTM network is validated better than the CNN model since its average and minimum F1 scores for validation are better than the CNN model (by 1~~-~~% and 32~~-~~%, respectively), and also LSTM yielded no F1 scores below 0.74 (Fig. 14 and Table~~-~~ 9).

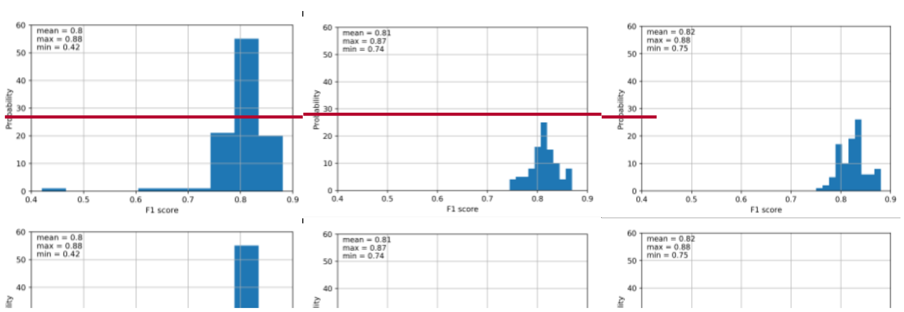
As shown in ~~Fig.~~ Figure 13, training loss is higher than validation loss in some of the results. There are some reasons explaining that. Regularization reduces the validation loss at the expense of increasing training loss. ~~The regularization~~Regularization techniques such as the application of noise layers are only ~~applied~~used during training, but not during validation resulting in ~~more smooth~~smoother and usually better functions in validation. There is no noise layer in CNN-LSTM model that ~~may cause~~could result in a lower training error than the validation error. However, other regularization methods such as L2 regularization are used in all the models, including the CNN-LSTM model.

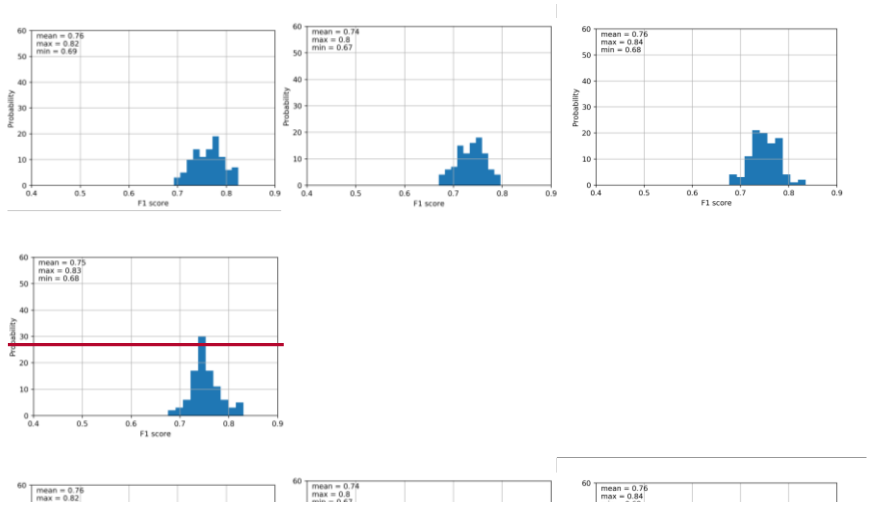
Furthermore, ~~the other issue is that~~ batch normalization uses the mean and variance of each batch ~~in~~during the training phase, whereas~~, in validation,~~ it uses the mean and variance of the whole training dataset~~. Plus~~ in the validation phase.

Additionally, training loss is averaged over each epoch, while validation losses are calculated ~~after~~upon completion of each ~~epoch once the current~~ training epoch~~ is completed~~. Hence, the training loss includes error calculations with fewer updates.

Among the developed machine learning models, SVM shows the best validation performance (Figure~~–~~ 15 and Table~~-~~ 10). However, F1 scores of deep learning models are much higher than those of machine learning models with an average of 6% higher F1 score resulted from CNN-LSTM model compared to the SVM model (Tables 9 and 10).







Table~~ ~~9. F1 scores of the validation phase for CNN, LSTM, and CNN-LSTM models with 100 random train-validation splits.

~~Models~~Model	F1 score		
	mean	max	min
CNN	0.80	0.88	0.42
LSTM	0.81	0.87	0.74
CNN-LSTM	0.82	0.88	0.75

Table~~ ~~10. F1 scores of the validation phase for SVM, DT, and KNN and MLP models with 100 random train-validation splits.

~~Models~~Model	F1 score		
	mean	max	min
SVM	0.76	0.82	0.69
DT	0.74	0.80	0.67
KNN	0.75	0.84	0.68
MLP	0.75	0.83	0.68

3.2.2 Number of parameters and run time

The total number of parameters in CNN, LSTM, and CNN-LSTM networks are 371586, 207874, and 664746, respectively. The CNN-LSTM model had the best performance ~~has resulted from CNN-LSTM~~ with the highest number of parameters. Even though the number of parameters for the LSTM model is less than that of the CNN model, the LSTM model shows better validation performance. Furthermore, the number of parameters in the CNN-LSTM model is much higher than the ~~two~~ other two models, ~~but the~~ without a large increase in computation time ~~is not much~~

Formatted: Font: Times New Roman, 10 pt
Formatted: Left
Formatted Table
Formatted: Font: Times New Roman, 10 pt
Formatted: Centered
Formatted: Font: Times New Roman, 10 pt
Formatted: Font: Times New Roman, 10 pt
Formatted: Font: Times New Roman, 10 pt
Formatted: Font: Times New Roman, 10 pt
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted Table
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman
Formatted: Font: Times New Roman

~~higher.~~. All three models ~~take less than~~ were trained within 24 hours ~~to train with~~using 100 shuffle splits for training and validation. The models ~~are~~were run on a CPU with four cores, 3.4~~-~~ GHz clock speed, and 12~~-~~ GB RAM. On the other hand, a few minutes were enough to train the machine learning models with 100 shuffle splits for training and validation. Although the training time for deep learning models is much higher than that of machine learning models, their superior performance, in this case, justifies their application.

~~For all the machine learning models, it took a couple of minutes to train with 100 shuffle splits for training and validation. Although, the training time for deep learning models is much higher than that of machine learning models, the much better performance of deep learning models justifies their application in our cases.~~

3.3 Order of input variables

It is not clear ~~that~~ whether the order of input variables in the input file ~~might influence~~influences multivariate TSC or not when using 2-D filters and 2-D max-pooling layers. In the benchmark~~, we randomly used this order~~ model, the variables were entered from left to right in the following random order: precipitation, minimum temperature, maximum temperature, net radiation, ATDD, AFDD, and snow depth. ~~We randomly changed this order and applied the new~~Another run was conducted by changing the order of the variables for the following random order: snow depth, maximum temperature, precipitation, AFDD, net radiation, minimum temperature, and ATDD. Both models yielded the same ~~average~~mean and minimum F1 scores, whereas the maximum F1 score ~~from the order in~~of the benchmark model ~~(0.88)~~ is higher (0.88) than that of the ~~second-order~~comparative run (0.86). Therefore, it can be concluded that the order does not significantly impact the results.

3.4 Testing

To examine the ability of the models to generalize to new unseen data, we randomly set aside 10% of the data from the training and validation ~~for~~phase of all the developed deep learning and machine learning models. ~~We trained a~~A CNN, an LSTM, and a CNN-LSTM model~~, then the~~ were trained ~~parameters are saved,~~ and ~~finally,~~ the well-trained parameters ~~are utilized for testing. We trained an~~were saved and used to assess the model's ability to generalize. An SVM, a DT, a KNN, and an MLP model ~~and the~~ were also trained. The trained models ~~are~~were saved and ~~later~~ used for testing. The test dataset is ~~almost a~~nearly balanced ~~dataset~~ with 101 samples with the size of (16, 7), including 48 ~~jams~~jam events and 53 no~~-jams~~ jam events.

The results of the test models show that the CNN-LSTM model ~~represent~~had the best F1 score of 0.92 (Table~~-~~ 11). Tables~~-~~ 9 and 11 show that although LSTM ~~has~~had a slightly ~~better~~superior validation performance, CNN and LSTM models performed the same in testing.

~~The~~Testing results of machine learning models ~~for testing~~are presented in Table~~-12 indicate that among~~ 11. Among the machine learning models, KNN yields the best results with F1 scores of 78%. ~~Tables 11 and 12 declare that deep learning models work much better than machine learning models for testing with 14%~~ By comparing ~~CNN-LSTM with KNN as~~ the best deep learning ~~and machine learning models,~~model (CNN-LSTM) with the best machine learning model (KNN), it can be calculated that the deep learning model outperforms the machine learning model by a difference of 14% (F1 score of 92% and 78%, respectively~~.~~).

Table~~ ~~ 11. Test F1 scores for ~~LSTM, CNN,~~the developed deep learning and ~~CNN-LSTM~~machine learning models.

Models	F1 score
CNN-LSTM	0.92
CNN	0.80
LSTM	0.80
~~CNN-LSTM~~KNN	0.~~92~~78

~~Table 12. Test F1 scores for SVM, DT, and KNN and MLP models.~~

~~Models~~	~~F1 score~~
SVM	0.75
DT	0.71
~~KNN~~	~~0.78~~
MLP	0.70

3.5 Model comparison

~~Multiple~~ Classifiers can be combined ~~classifiers can be considered for~~ and used in pattern recognition problems to reduce errors ~~as different classifiers can cover~~by covering for one another's internal weaknesses ~~of each other~~ (Parvin et al., 2011). ~~The combined classifier~~ Combined classifiers may be less accurate than the most accurate classifier~~.~~ ~~However~~, however, the accuracy of the combined model is always ~~higher than~~superior to the average accuracy of individual models. Combining two models improved our results compared to convolution-only or LSTM-only networks in both training and testing, supporting the previous studies (e.g., Sainath et al., 2015). It can be because the CNN-LSTM model incorporates both the temporal dependency of each variable by using LSTM networks and the correlation between variables through CNN models. The combined CNN-LSTM model efficiently benefit from automatic feature learning by CNN plus the native support for time series by LSTM.

Although LSTM ~~performed~~ slightly ~~better than~~outperformed CNN in the validation phase, these models ~~showed the same~~had comparable performance in the testing phase. The CNN is able to partially include both temporal dependency and the correlation between variables by using ~~1D~~1-D and ~~2D~~2-D filters, respectively. Although the LSTM is unable to incorporate the correlations between variables, it gives promising results with relatively small dataset~~ and~~. Another difference is that LSTM captures longer temporal dynamics, while the CNN only captures temporal dynamics comprised within the length of its filters.

Even though our training data in supervised ice-jam prediction is small, the results reveal that deep learning techniques can give accurate results, which agrees with a previous study conducted by Ordóñez and Roggen (2016) in activity recognition. The excellent performance of CNN and CNN-LSTM models may be partially due to the characteristic of CNN that decreases the total number of parameters which does training with limited training data easier (Gao et al., 2016~~, May~~). However, we expect our models ~~will~~to be improved in the future by a larger dataset.

Formatted: Font: Times New Roman, 10 pt

Formatted: Font: Times New Roman, 10 pt

Formatted Table

Formatted: Font: Times New Roman, 10 pt

Formatted: Font: Times New Roman, 10 pt

Formatted: Font: Times New Roman, 10 pt, Font color: Auto

Formatted: Font: Times New Roman, 10 pt

Formatted: Font: Times New Roman, 10 pt

Formatted Table

Formatted: Font: 10 pt

Formatted: Font: 10 pt, Font color: Black

Formatted: Font: Times New Roman, 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt, Font color: Black

Formatted: Font: Times New Roman, 10 pt

Formatted Table

Formatted: Font: 10 pt

Formatted: Font: 10 pt, Font color: Black

Among the developed machine learning models, SVM showed the best performance in validation, whereas KNN worked the best in testing. However, the performance of deep learning models is much better than machine learning models in both validation and testing. The machine learning models do not consider correlations between variables. However, it is not the only reason that deep learning models worked better than machine learning models. As the LSTM also does not consider correlations between variables but worked better than machine learning models. ~~Some characteristics~~This indicates that there are other aspects of ~~developed~~ deep learning models ~~can explain~~ that contribute to their ~~better~~high performance ~~compared to machine learning models~~level. For instance, deep learning models perform well for the problems with complex-nonlinear dependencies, time dependencies, and multivariate inputs.

The developed CNN-LSTM model can be used for future predictions of ice jams in Quebec to provide early warning of possible floods in the area by using historic hydro-meteorological variables and their predictions for some days in advance.

3.6 Discussion on the interpretability of deep learning models

Even though the developed deep learning models performed ~~pretty~~ well in predicting ice jams in Quebec, the interpretability of the results with respect to the physical processes ~~of the~~involved in ice ~~jam~~jams is still essential. It is because although deep learning models have achieved superior performance in various tasks, these ~~really~~ complicated models ~~with~~use a large number of parameters and might sometimes exhibit unexpected behaviours (Samek et al., 2017 ~~&~~; Zhang et al., 2021). This is because the real-world environment is still much more complex than any model. Furthermore, the models may learn some spurious correlations in the data and make correct predictions ~~with~~for the ~~'wrong'~~"wrong" reason (Samek and Müller, 2019). Hence, interpretability is especially important in some real-world applications like flood and ice-jam predictions where an error ~~may cause~~could have catastrophic ~~results. Also~~consequences. Nonetheless, interpretability can be used to extract novel domain knowledge and hidden laws of nature in the research fields with limited domain knowledge (Alipanahi et al., 2015) like ice-jam prediction.

However, the nested ~~non-linear~~nonlinear structure and the "black box" nature of deep neural networks make ~~interpretability~~the interpretation of their underlying mechanisms and ~~their~~ decisions a significant challenge (Montavon et al., 2018~~,~~; Zhang et al., 2021 ~~and~~; Wojtas and Chen, 2020). That is why, ~~interpretability of~~ deep neural ~~networks~~network interpretability still remains a young and emerging field of research. Nevertheless, there are various methods available to facilitate the understanding of decisions made by a deep learning model such as feature importance ranking, sensitivity analysis, layer-wise relevance propagation, and the global surrogate model. However, the interpretability of developed deep learning models for ice-jam prediction is beyond the scope of this study and it will be investigated in our future works.

3.7 Model transferability

The transferability of a model between river basins is highly desirable but has not yet been achieved because most river ice-jam models are site specific (Mahabir et al., 2007). The developed models in this study can be used to predict future ice jams some days before the event not only for Quebec but can also ~~for~~be transferred to eastern parts of

Ontario and western New Brunswick~~.~~, since these areas have the similar hydro-meteorological conditions. For other locations, the developed models ~~can~~could be ~~transferred via re-training and~~retrained with a small amount of fine-tuning using ~~labeled~~labelled instances~~,~~ rather than building from scratch. ~~It~~This interesting feature is ~~because~~due to the logic ~~in~~behind the model~~ may~~, which could be ~~transferable~~transferred to~~ the~~ other sites with small modifications. To transfer a model from one river basin to another, ~~historic~~historical records of ice jams and equivalent hydro-meteorological variables (e.g., precipitation, temperature, and snow depth) ~~as inputs to the model~~ must be available ~~at~~as model inputs for each site.

4 Conclusion

The main finding from this project is that ~~all~~ the developed deep models ~~performed pretty well~~successfully predicted ice jams in Quebec, and performed much better than the developed machine learning models~~ for ice-jam prediction in Quebec. The comparison of~~. The results show that the CNN-LSTM model is superior to the CNN-only and LSTM-only networks in both validation and testing ~~accuracy, though~~phases, although the LSTM and CNN ~~models demonstrate quite good performance~~performed well.

To our best knowledge, this study is the first ~~study introducing these~~to apply deep learning models to the ~~problem of~~ ice-jam prediction. The developed models are promising ~~to be used to predict future~~tools for the prediction of ice jams in Quebec and ~~in~~ other similar river basins in Canada with re-training and a small amount of fine-tuning.

The developed models do not apply to freeze-up jams that occur in early winter and are based on different processes than breakup jams. We studied only breakup ice jams as usually they result in flooding and are more dangerous than freeze-up jams. ~~-~~Furthermore, there is a lack of data availability for freeze-up ice jams in Quebec and only 89 records of freeze-up jams are available which is too small.

The main limitation of this study is ~~data~~the availability ~~as recorded~~of ice ~~jams are~~jam records. Indeed, small ~~which causes~~datasets may lead deep learning models to ~~easily~~ overfit ~~to small number of~~ the data. Another limitation of the presented work is the lack of interpretability of the results with respect to the physical characteristics of the ice jam. This is a topic of future research and our next step is to explore that.

~~The~~It should also be noted that hydro-meteorological variables are not the only drivers of ice-jam formation. ~~The geomorphological indicators that control the formation of ice jams include the~~ Geomorphological features such as river slope, sinuosity, ~~a barrier~~ physical barriers (such as an island or a bridge~~,~~), channel narrowing~~ of the channel~~, and river confluence ~~of rivers~~also govern the formation of ice jams. In the future, a geospatial model using deep learning will be developed to examine the impacts of these geospatial parameters on ice-jam formation.

Author contribution

Fatemehalsadat Madaeni designed and carried out the experiments under Karem Chokmani and Saeid Homayouni supervision. Fatemehalsadat Madaeni developed the model code and performed the simulations using hydro-meteorological and ice-jam data provided and validated by Rachid Lhissou. Fatemehalsadat Madaeni wrote the bulk of the paper with conceptual edits from Karem Chokmani and Saeid Homayouni. Yves Gauthier and Simon Tolszczuk-Leclerc helped in the refinement of the objectives and the revision of the methodological developments.

Acknowledgment

This study is part of the DAVE project, funded by the Defence Research and Development Canada (DRDC), Canadian Safety and Security Program (CSSP), with partners from Natural Resources Canada (NRCan), and Environment and Climate Change Canada.

References

Alipanahi, B., Delong, A., Weirauch, M. T., & Frey, B. J. (2015). Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. Nature biotechnology, 33(8), 831-838.

Althoff, D., Rodrigues, L. N., & Bazame, H. C. (2021). Uncertainty quantification for hydrological models based on neural networks: the dropout ensemble. Stochastic Environmental Research and Risk Assessment, 35(5), 1051-1067.

Analytics, C. (2016). Anaconda Software Distribution: Version 2-2.4. 0.

Apaydin, H., Feizi, H., Sattari, M. T., Colak, M. S., Shamshirband, S., & Chau, K. W. (2020). Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting. Water, 12(5), 1500.

Barnes-Svarney, P. L., & Montz, B. E. (1985). An ice jam prediction model as a tool in floodplain management. Water Resources Research, 21(2), 256-260.

Barzegar, R., Aalami, M. T., & Adamowski, J. (2020). Short-term water quality variable prediction using a hybrid CNN–LSTM deep learning model. Stochastic Environmental Research and Risk Assessment, 1-19.

Barzegar, R., Aalami, M. T., & Adamowski, J. (2021). Coupling a hybrid CNN-LSTM deep learning model with a Boundary Corrected Maximal Overlap Discrete Wavelet Transform for multiscale Lake water level forecasting. Journal of Hydrology, 598, 126196.

Beltaos, S. (1993). Numerical computation of river ice jams. Canadian Journal of Civil Engineering, 20(1), 88-99.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. Neural networks: Tricks of the trade, 437-478.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(2).

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., ... & Bengio, Y. (2010~~, June~~). Theano: A CPU and GPU math compiler in Python. ~~In~~ Proc. 9th Python in Science Conf~~ (Vol. 1, pp.~~, 3-10~~).~~.

~~Brownlee, J. (2018).~~Brownlee, J. (2017). Long short-term memory networks with python: develop sequence prediction models with deep learning. Machine Learning Mastery.

Brownlee, J. (2018a). Better deep learning: train faster, reduce overfitting, and make better predictions. Machine Learning Mastery.

Formatted: French (Canada)

Brownlee, J. (2018b). Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery.

Brunel, A., Pasquet, J., PASQUET, J., Rodriguez, N., Comby, F., Fouchez, D., & Chaumont, M. (2019). A CNN adapted to time series for the classification of Supernovae. Electronic Imaging, 2019(14), 90-1.

Brunello, A., Marzano, E., Montanari, A., & Sciavicco, G. (2019). J48SS: A novel decision tree approach for the handling of sequential and time series data. Computers, 8(1), 21.

Brunner, G. W. (2002). Hec-ras (river analysis system).~~ In~~ North American Water and Environment Congress & Destructive Water~~ (pp.,~~ 3782-3787~~). ASCE~~.

Carson, R. W., Beltaos, S., Healy, D., & Groeneveld, J. (2003~~, June~~). Tests of river ice jam models–phase 2. ~~In ~~Proceedings of the 12th Workshop on the Hydraulics of Ice Covered Rivers, Edmonton, Alta~~ (pp.,~~ 19-20~~)~~.

Carson, R., Beltaos, S., Groeneveld, J., Healy, D., She, Y., Malenchak, J., ... & Shen, H. T. (2011). Comparative testing of numerical models of river ice jams. Canadian Journal of Civil Engineering, 38(6), 669-678.

Chen, R., Wang, X., Zhang, W., Zhu, X., Li, A., & Yang, C. (2019). A hybrid CNN-LSTM model for typhoon formation forecasting. GeoInformatica, 23(3), 375-396.

Cui, Z., Chen, W., & Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. arXiv preprint arXiv:1603.06995.

De Coste, M., Li, Z., Pupek, D., & Sun, W. (2021). A hybrid ensemble modelling framework for the prediction of breakup ice jams on Northern Canadian Rivers. Cold Regions Science and Technology, 189, 103302.

del Campo, F. A., Neri, M. C. G., Villegas, O. O. V., Sánchez, V. G. C., Domínguez, H. D. J. O., & Jiménez, V. G. (2021). Auto-adaptive multilayer perceptron for univariate time series classification. Expert Systems with Applications, 181, 115147.

Devineau, G., Moutarde, F., Xi, W., & Yang, J. (~~2018, May~~2018a). Deep learning for hand gesture recognition on skeletal data. ~~In 2018 ~~13th IEEE International Conference on Automatic Face & Gesture Recognition~~ (FG 2018) (pp.,~~ 106-113~~). IEEE~~.

Devineau, G., Xi, W., Moutarde, F., & Yang, J. (~~2018, June~~2018b). Convolutional neural networks for multivariate time series classification using both inter-and intra-channel parallel convolutions. ~~In ~~Reconnaissance des Formes, Image, Apprentissage et Perception~~ (RFIAP'2018)~~.

Dieleman, S., Schlüter, J., Raffel, C., Olson, E., Sønderby, S.K., Nouri, D., … & Degrave,J. (2015). Lasagne: First release. (Version v0.1). Zenodo. Retrieved from http://doi.org/10.5281/zenodo.27878.

Formatted: Font: Not Italic

Formatted: Font: Not Italic

Formatted: French (Canada)

Formatted: Font: Not Italic

Formatted: Font: Not Italic

Formatted: English (Canada)

Formatted: English (Canada)

Données Québec: Historique (publique) d'embâcles répertoriés au MSP - Données Québec~~,~~. Retrieved from https://www.donneesquebec.ca/recherche/dataset/historique-publique-d-embacles-repertories-au-msp. (~~last~~Last access: 15 June 2021).

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. A. (~~2019, July~~2019a). Deep neural network ensembles for time series classification. ~~In 2019~~ International Joint Conference on Neural Networks~~ (IJCNN) (pp.~~, 1-6~~). IEEE~~.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. A. (~~2019~~2019b). Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, 33(4), 917-963.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.

Gao, Y., Hendricks, L. A., Kuchenbecker, K. J., & Darrell, T. (2016~~, May~~). Deep learning for tactile understanding from visual and haptic data. ~~In 2016 IEEE~~ International ~~-~~Conference on Robotics and Automation~~ (ICRA) (pp.~~, 536-543~~). IEEE~~.

Garbin, C., Zhu, X., & Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. Multimedia Tools and Applications, 1-39.

Glorot, X., & Bengio, Y. (2010~~, March~~). Understanding the difficulty of training deep feedforward neural networks. ~~In~~ Proceedings of the ~~thirteenth international conference~~ Thirteenth International Conference on ~~artificial intelligence~~Artificial Intelligence and ~~statistics (pp.~~Statistics, 249-256~~). JMLR Workshop and Conference Proceedings~~.

Graf, R., Kolerski, T., & Zhu, S. (2022). Predicting Ice Phenomena in a River Using the Artificial Neural Network and Extreme Gradient Boosting. Resources, 11(2), 12.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. Pattern Recognition, 77, 354-377.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

Hatami, N., Gavet, Y., & Debayle, J. (2018~~, April~~). Classification of time-series images using deep convolutional neural networks. ~~In~~ Tenth International Conference on Machine Vision~~ (ICMV 2017) (Vol. 10696, p. 106960Y). International Society for Optics and Photonics~~.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. ~~In~~ Proceedings of the IEEE ~~international conference~~International Conference on ~~computer vision (pp.~~Computer Vision, 1026-1034~~)~~.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. IEEE Annals of the History of Computing, 9(03), 90-95.

Ioffe, S., & Szegedy, C. (2015~~, June~~). Batch normalization: Accelerating deep network training by reducing internal covariate shift. ~~In~~ International ~~conference~~Conference on ~~machine learning (pp.~~Machine Learning, 448-456~~). PMLR~~.

Jović, A., Brkić, K., & Bogunović, N. (2012~~, August~~). Decision tree ensembles in biomedical time-series classification. ~~In~~ Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium ~~(pp.~~, 408-417~~). Springer, Berlin, Heidelberg.~~.

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015~~, June~~). An empirical exploration of recurrent network architectures. ~~In~~ International ~~conference~~Conference on ~~machine learning (pp.~~Machine Learning, 2342-2350~~). PMLR~~.

Karim, F., Majumdar, S., & Darabi, H. (2019a). Insights into LSTM fully convolutional networks for time series classification. IEEE Access, 7, 67718-67725.

Karim, F., Majumdar, S., Darabi, H., & Chen, S. (2017). LSTM fully convolutional networks for time series classification. IEEE access, 6, 1662-1669.

Karim, F., Majumdar, S., Darabi, H., & Harford, S. (~~2019~~2019b). Multivariate lstm-fcns for time series classification. Neural Networks, 116, 237-245.

Kashiparekh, K., Narwariya, J., Malhotra, P., Vig, L., & Shroff, G. (2019~~, July~~). ConvTimeNet: A pre-trained deep convolutional neural network for time series classification. ~~In 2019~~ International Joint Conference on Neural Networks ~~(IJCNN) (pp.~~, 1-8~~). IEEE~~.

Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018). Rainfall–runoff modelling using long short-term memory (LSTM) networks. Hydrology and Earth System Sciences, 22(11), 6005-6022.

Li, D., Djulovic, A., & Xu, J. F. (2013). A Study of kNN using ICU multivariate time series data. ~~In~~ Proc. Int. Conf. Data Mining, eds. R. Stahlbock and GM Weiss ~~(DMIN, 2013) (pp.~~, 211-217~~).~~.

Li, X., Zhang, Y., Zhang, J., Chen, S., Marsic, I., Farneth, R. A., & Burd, R. S. (2017). Concurrent activity recognition with multimodal CNN-LSTM structure. arXiv preprint arXiv:1702.01638.

Lin, J., Williamson, S., Borne, K., & DeBarr, D. (2012). Pattern recognition in time series. Advances in Machine Learning and Data Mining for Astronomy, 1, 617-645.

Lindenschmidt, K. E. (2017). RIVICE—a non-proprietary, open-source, one-dimensional river-ice model. Water, 9(5), 314.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019.

Formatted: French (Canada)

Formatted: Font: Not Italic

Formatted: Font: Not Italic

Livieris, I. E., Pintelas, E., & Pintelas, P. (2020). A CNN–LSTM model for gold price time-series forecasting. Neural computing and applications, 32(23), 17351-17360.

Lu, N., Wu, Y., Feng, L., & Song, J. (2018). Deep learning for fall detection: Three-dimensional CNN combined with LSTM on video kinematic data. IEEE journal of biomedical and health informatics, 23(1), 314-323.

Luan, Y., & Lin, S. (2019~~, March~~). Research on text classification based on CNN and LSTM. ~~In 2019 IEEE international conference~~International Conference on ~~artificial intelligence~~Artificial Intelligence and ~~computer applications (ICAICA) (pp.~~Computer Applications, 352-355~~). IEEE~~.

Madaeni, F., Lhissou, R., Chokmani, K., Raymond, S., & Gauthier, Y. (2020). Ice jam formation, breakup and prediction methods based on hydroclimatic data using artificial intelligence: A review. Cold Regions Science and Technology, 103032.

Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2017). High-resolution aerial image labeling with convolutional neural networks. IEEE Transactions on Geoscience and Remote Sensing, 55(12), 7092-7103.

Mahabir, C., Hicks, F. E., & Fayek, A. R. (2007). Transferability of a neuro-fuzzy river ice jam flood forecasting model. Cold Regions Science and Technology, 48(3), 188-201.

Mahabir, C., Hicks, F., & Fayek, A. R. (2006). Neuro-fuzzy river ice breakup forecasting system. Cold regions science and technology, 46(2), 100-112.

Mahfouf, J. F., Brasnett, B., & Gagnon, S. (2007). A Canadian precipitation analysis (CaPA) project: Description and preliminary results. Atmosphere-ocean, 45(1), 1-17.

Massie, D.D., White, K.D., Daly, S.F.~~, .~~ (2002~~.~~). Application of neural networks to predict ice jam occurrence. Cold ~~Reg. Sci. Technol.~~Regions Science and Technology, 35 (2), 115–122.

Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks. arXiv preprint arXiv:1804.07612.

Mesinger, F., DiMego, G., Kalnay, E., Mitchell, K., Shafran, P. C., Ebisuzaki, W., ... & Shi, W. (2006). North American regional reanalysis. Bulletin of the American Meteorological Society, 87(3), 343-360.

Montavon, G., Samek, W., & Müller, K. R. (2018). Methods for interpreting and understanding deep neural networks. Digital Signal Processing, 73, 1-15.

Mutegeki, R., & Han, D. S. (2020~~, February~~). A CNN-LSTM approach to human activity recognition. ~~In 2020~~ International Conference on Artificial Intelligence in Information and Communication~~ (ICAIIC) (pp.~~, 362-366~~). IEEE~~.

Nanopoulos, A., Alcock, R., & Manolopoulos, Y. (2001). Feature-based classification of time-series data. International Journal of Computer Research, 10(3), 49-61.

National Hydro Network - NHN - GeoBase Series - Natural Resources Canada. Retrieved from https://open.canada.ca/data/en/dataset/a4b190fe-e090-4e6d-881e-b87956c07977.

National Hydrographic Network - Natural Resources Canada. Retrieved from https://www.nrcan.gc.ca/science-and-data/science-and-research/earth-sciences/geography/topographic-information/geobase-surface-water-program-geeau/national-hydrographic-network/21361.

Nosratabadi, S., Mosavi, A., Duan, P., Ghamisi, P., Filip, F., Band, S. S., ... & Gandomi, A. H. (2020). Data science in economics: comprehensive review of advanced machine learning and deep learning methods. Mathematics, 8(10), 1799.

Oh, S. L., Ng, E. Y., San Tan, R., & Acharya, U. R. (2018). Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats. Computers in biology and medicine, 102, 278-287.

Olah, C. (2015). Understanding LSTM Networks. Retrieved~~—~~ from https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Ombabi, A. H., Ouarda, W., & Alimi, A. M. (2020). Deep learning CNN–LSTM framework for Arabic sentiment analysis using textual information shared in social networks. Social Network Analysis and Mining, 10(1), 1-13.

Ordóñez, F. J., & Roggen, D. (2016). Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors, 16(1), 115.

Parvin, H., Minaei, B., Beigi, A., & Helmi, H. (2011~~, April~~). Classification ensemble by genetic algorithms. ~~In~~ International Conference on Adaptive and Natural Computing Algorithms~~ (pp.~~, 391-399~~). Springer, Berlin, Heidelberg.~~.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. ~~the~~The Journal of ~~machine~~Machine Learning ~~research~~Research, 12.

Prowse, T. D., & Bonsal, B. R. (2004). Historical trends in river-ice break-up: a review. Hydrology Research, 35 (4-5), 281-293.

Prowse, T. D., Bonsal, B. R., Duguay, C. R., & Lacroix, M. P. (2007). River-ice break-up/freeze-up: a review of climatic drivers, historical trends and future predictions. Annals of Glaciology, 46, 443-451.

Raybaut, P. (2009). Spyder-documentation. Retrieved from pythonhosted. org.

Reback, J., McKinney, W., Den Van Bossche, J., Augspurger, T., Cloud, P., Klein, A., ... & Seabold, S. (2020). pandas-dev/pandas: Pandas 1.0. 3. Zenodo.

Rodríguez, J. J., & Alonso, C. J. (2004~~, December~~). Support vector machines of interval-based features for time series classification. ~~In~~ International Conference on Innovative Techniques and Applications of Artificial Intelligence~~ (pp.~~, 244-257~~). Springer, London~~.

Formatted: French (Canada)

Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015~~, April~~). Convolutional, long short-term memory, fully connected deep neural networks. ~~In 2015 IEEE~~ International Conference on Acoustics, Speech and Signal Processing ~~(ICASSP) (pp.,~~ 4580-4584~~). IEEE~~.

Samek, W., & Müller, K. R. (2019). Towards explainable artificial intelligence. ~~In~~ Explainable AI: ~~interpreting, explaining~~Interpreting, Explaining and ~~visualizing deep learning (pp.~~Visualizing Deep Learning, 5-22~~). Springer, Cham~~.

Samek, W., Wiegand, T., & Müller, K. R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296.

Sarafanov, M., Borisova, Y., Maslyaev, M., Revin, I., Maximov, G., & Nikitin, N. O. (2021). Short-Term River Flood Forecasting Using Composite Models and Automated Machine Learning: The Case Study of Lena River. Water, 13(24), 3482.

Semenova, N., Sazonov, A., Krylenko, I., & Frolova, N. (2020). Use of classification algorithms for the ice jams forecasting problem. E3S Web of Conferences.

She, X., & Zhang, D. (2018~~, December~~). Text classification based on hybrid CNN-LSTM hybrid model. ~~In 2018~~ 11th International Symposium on Computational Intelligence and Design ~~(ISCID) (Vol. 2, pp.,~~ 185-189~~). IEEE~~.

Shouyu, C., & Honglan, J. (2005). Fuzzy Optimization Neural Network Approach for Ice Forecast in the Inner Mongolia Reach of the Yellow River/Approche d'Optimisation Floue de Réseau de Neurones pour la Prévision de la Glace Dans le Tronçon de Mongolie Intérieure du Fleuve Jaune. Hydrological sciences journal, 50(2).

Sosa, P. M. (2017). Twitter sentiment analysis using combined LSTM-CNN models. Eprint Arxiv, 1-9.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

The Atlas of Canada - Toporama - Natural Resources Canada. Retrieved from https://atlas.gc.ca/toporama/en/index.html.

Thornton, M.M., Shrestha, R., Wei, Y., Thornton, P.E., Kao, S. & Wilson, B.E. (2020). Daymet: Daily Surface Weather Data on a 1-km Grid for North America, Version 4. ORNL DAAC, Oak Ridge, Tennessee, USA.

Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., & Troncoso, A. (2021). Deep Learning for Time Series Forecasting: A Survey. Big Data, 9(1), 3-21.

Turcotte, B., & Morse, B. (2015~~, August~~). River ice breakup forecast and annual risk distribution in a climate change perspective. ~~In~~ 18th Workshop on the Hydraulics of Ice Covered Rivers, CGU HS Committee on River Ice Processes and the Environment, Quebec ~~(Vol. 35).~~.

Formatted: English (Canada)

Umer, M., Imtiaz, Z., Ullah, S., Mehmood, A., Choi, G. S., & On, B. W. (2020). Fake news stance detection using deep learning architecture (cnn-lstm). IEEE Access, 8, 156695-156706.

Wang, J., Yu, L. C., Lai, K. R., & Zhang, X. (2016~~, August~~). Dimensional sentiment analysis using a regional CNN-LSTM model. ~~In~~ Proceedings of the 54th ~~annual meeting~~Annual Meeting of the ~~association~~Association for ~~computational linguistics (volume 2: Short papers) (pp.~~Computational Linguistics, 225-230~~)~~.

Wang, J., Yu, L. C., Lai, K. R., & Zhang, X. (2019). Tree-structured regional CNN-LSTM model for dimensional sentiment analysis. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 28, 581-591.

White, K. D. (2003). Review of prediction methods for breakup ice jams. Canadian Journal of Civil Engineering, 30(1), 89-100.

White, K. D., & Daly, S. F. (2002~~, January~~). Predicting ice jams with discriminant function analysis. ~~In ASME 2002~~ 21st International Conference on Offshore Mechanics and Arctic Engineering ~~(pp.~~, 683-690~~). American Society of Mechanical Engineers~~.

Wojtas, M., & Chen, K. (2020). Feature importance ranking for deep learning. arXiv preprint arXiv:2010.08973.

Wu, J., Yao, L., & Liu, B. (~~2018a, April~~2018). An overview on feature-based classification algorithms for multivariate time series. ~~In 2018 IEEE~~ 3rd International Conference on Cloud Computing and Big Data Analysis ~~(ICCCBDA) (pp.~~, 32-38~~). IEEE~~.

Wu, Z., Wang, X., Jiang, Y. G., Ye, H., & Xue, X. (2015~~, October~~). Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. ~~In Proceedings of the~~ 23rd ACM ~~international conference~~International Conference on Multimedia ~~(pp.~~, 461-470~~)~~.

Wunsch, A., Liesch, T., & Broda, S. (2020). Groundwater Level Forecasting with Artificial Neural Networks: A Comparison of LSTM, CNN and NARX. Hydrology and Earth System Sciences Discussions, 2020, 1-23.

Xing, Z., Pei, J., & Keogh, E. (2010). A brief survey on sequence classification. ACM Sigkdd Explorations Newsletter, 12(1), 40-48.

Xingjian, S. H. I., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. ~~In~~ Advances in neural information processing systems ~~(pp.~~, 802-810~~)~~.

Yan, J., Mu, L., Wang, L., Ranjan, R., & Zomaya, A. Y. (2020). Temporal convolutional networks for the advance prediction of ENSO. Scientific reports, 10(1), 1-15.

Yang, J., Nguyen, M. N., San, P. P., Li, X. L., & Krishnaswamy, S. (2015~~, June~~). Deep convolutional neural networks on multichannel time series for human activity recognition. ~~In~~ Twenty-fourth ~~international joint conference~~International Joint Conference on ~~artificial intelligence~~Artificial Intelligence.

Yi, S., Ju, J., Yoon, M. K., & Choi, J. (2017). Grouped convolutional neural networks for multivariate time series. arXiv preprint arXiv:1703.09938.

Zhang, D., Lin, J., Peng, Q., Wang, D., Yang, T., Sorooshian, S., ... & Zhuang, J. (2018). Modeling and simulating of reservoir operation using the artificial neural network, support vector regression, deep learning algorithm. Journal of Hydrology, 565, 720-736.

Zhang, Y., Tiňo, P., Leonardis, A., & Tang, K. (2021). A survey on neural network interpretability. IEEE Transactions on Emerging Topics in Computational Intelligence.

Zhao, L., Hicks, F. E., & Fayek, A. R. (2012). Applicability of multilayer feed-forward neural networks to model the onset of river breakup. Cold Regions Science and Technology, 70, 32-42.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2014~~, June~~). Time series classification using multi-channels deep convolutional neural networks. ~~In~~ International Conference on Web-Age Information Management~~ (pp.~~, 298-310~~). Springer, Cham.~~.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2016). Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. Frontiers of Computer Science, 10(1), 96-112.