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

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 result in severe flooding due to a rapid rise in water levels upstream of the jam. These floods threaten human safety and damage properties and infrastructures as the floods resulting from ice-jams are sudden. Hence, ~~the~~ ice-jam prediction tools can give an early warning to increase response time and minimize the possible corresponding damages. However, ~~the~~ ice-jam prediction has always been a challenging problem 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. ~~The ice~~Ice-jam prediction problem can be considered as a binary multivariate time-series classification. Deep learning techniques have been ~~successfully applied~~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 (CNN~,~), Long Short-Term Memory (LSTM~,~), and combined ~~CN~~Convolutional-Long Short-Term Memory (CNN-LSTM) networks for ice-jam prediction for ~~all the~~150 rivers in Quebec. The hydro-meteorological variables (e.g., temperature, precipitation, and snow depth) along with the corresponding jam or no-jam events are used as the inputs to the models. We hold out 10% of the data for testing. And we applied 100 re-shuffling and splitting iterations with 80 % of the remaining data for training and 20% for validation. The results show that the ~~CN~~CNN-LSTM model yields the best results in the validation and ~~generalization~~testing with F1 scores of 0.82 and 0.~~91~~92, respectively. This demonstrates that CNN and LSTM models are complementary, and a combination of them further improves classification.

1 Introduction

Predicting ice-jam events gives an early warning of possible flooding, but there is no analytical solution to predict these events due to the complex interactions between involved hydro-meteorological variables~.~ (e.g., temperature, precipitation, snow depth, and solar radiation). To date, a small number of empirical and statistical prediction methods ~~that have been developed~~ (such as threshold methods, multi-regression models, logistic regression models, and discriminant function analysis) ~~for ice jams~~ 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 high rate 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 limitations in predicting ice-jam occurrence. This is because mathematical formulations in these models are complex which need many parameters that are often unavailable as they are challenging to measure in ice conditions. Hence,

38 many simplifications corresponding to these parameters may degrade model accuracy (Shouyu & Honglan, 2005). A

39 detailed overview of the previous models for ice-jam prediction based on hydro-meteorological data are presented in

40 Madaeni et al. (2020).

41 Prediction of ice-jam occurrence can be considered as a binary multivariate time-series classification (TSC)

42 ~~model~~problem when the time series of various hydro-meteorological variables (explained later) can be used to classify

43 ~~to~~ jam or no jam events. Time-series classification (particularly multivariate) has been widely used in various fields,

44 including biomedical engineering, clinical prediction, human activity recognition, weather forecasting, and finance.

45 Multivariate time-series provide more patterns and improve classification performance compared to univariate time-

46 series (Zheng et al., 2016). Time-series classification is one of the most challenging problems in data mining and

47 machine learning.

48 Most existing TSC methods are feature-based, distance-based, or ensemble methods (Cui et al., 2016). Feature

49 extraction is challenging due to the difficulty of handcrafting useful features to capture intrinsic characteristics from

50 time-series data (Karim et al., 2019; Zheng et al., 2014, June). Hence, distance-based methods work better in TSC

51 (Zheng et al., 2014, June). Among the hundreds of methods developed ~~methods~~ for TSC, the leading classifier with

52 the best performance was ensemble nearest neighbor with dynamic time warping (DTW) for many years (Fawaz et

53 al., 2019, July; Karim et al., 2019).

54 In the k-nearest neighbors (~~kNN~~KNN) classifier, the given test instance is classified by a majority vote of its k closest

55 neighbors in the training data. The ~~kNN~~KNN classifier needs all the data to make a prediction which requires high

56 memory. Hence, it is computationally expensive and could be slow if the database is large, and sensitive to irrelevant

57 features and the scale of the data. Furthermore, the number of neighbors to include in the algorithm should be

58 ~~wisely~~carefully selected. The ~~kNN~~KNN classifier is very challenging to be used for multivariate TSC. The dynamic

59 time warping is a more robust alternative for Euclidean distance (the most widely used time-series distance measure)

60 to measure the similarity between two given time series by searching for an optimal alignment (minimum distance)

61 between them (Zheng et al., 2016). However, the combined ~~kNN~~KNN with DTW is time-consuming and inefficient

62 for long multivariate time-series (Lin et al., 2012; Zheng et al., 2014, June). The traditional classification and classic

63 data mining algorithms developed for TSC have high computational complexity or low prediction accuracy. This is

64 due to the size and inherent complexity of time series, seasonality, noise, and feature correlation (Lin et al., 2012).

65 There are some machine learning methods available for TSC such as KNN and support vector machine (SVM).

66 However, the focus of this research is on the deep learning models that have greatly impacted sequence classification

67 problems and they can also be used for multivariate TSC with good performance. Deep learning methods are able to

68 consider two-dimensionality in multivariate time-series and their deeper architecture could further improve the

69 classification especially for complex problems, which is why their results are more accurate and robust than other

70 methods (Wu et al., 2018a, April). However, they are more time consuming and difficult to interpret.

71 Deep learning is a type of neural ~~network~~networks that uses multiple layers ~~of~~where nonlinear

72 ~~information~~transformation is used to extract

73 higher-level features from the input data. Although deep learning in recent years showed promising performance in

74 various fields such as image and speech recognition, document classification, and natural language processing, only a

75   few studies employed deep learning for TSC (Gu et al., 2018; Fawaz et al., 2019, July). Various studies show that

76   deep neural networks significantly outperform the ensemble nearest neighbor with DTW (Fawaz et al., 2019, July).

77   The main benefit of deep learning networks is automatic feature-extraction, which reduces the need for expert

78   knowledge of the field and removes engineering bias in the classification task (Fawaz et al., 2019) as the probabilistic

79   decision (e.g., classification) is taken by the network.

80   The most widely used deep neural networks for TSC are Multi-Layer Perceptron (MLP; i.e., fully connected deep

81   neural networks), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM~~).~~

82   ). The application of CNNs for TSC has recently become more and more popular, and different types of CNN are

83   being developed with superior accuracy ~~performance~~ for this purpose (e.g., Cui et al., 2016). Zheng et al. (2014, June)

84   and Zheng et al. (2016) introduce a Multi-Channels Deep Convolutional Neural Network (MC-DCNN) for

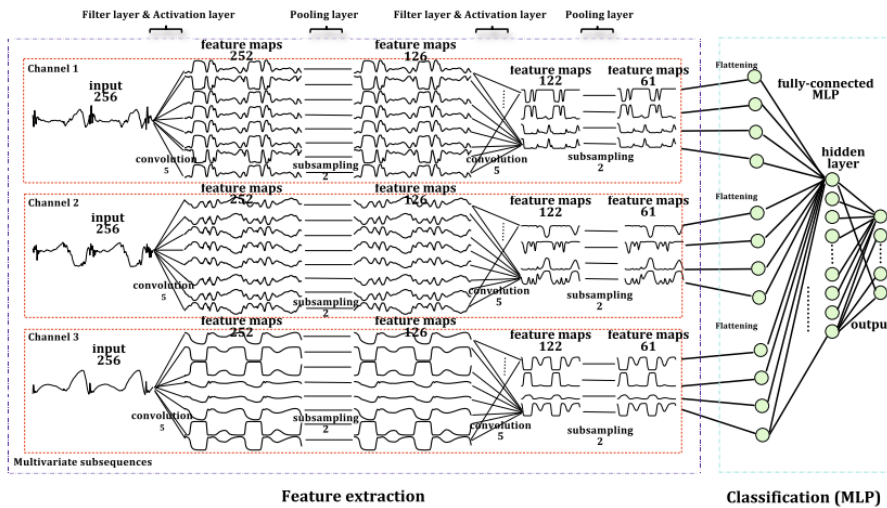
85   multivariate TSC, where each variable (i.e., univariate time series) is trained individually to extract features and finally

86   concatenated using an MLP to perform classification (Fig. 1). ~~Their results show~~They showed that their model

87   achieves a state-of-the-art performance both in efficiency and accuracy on a challenging dataset. The drawback of

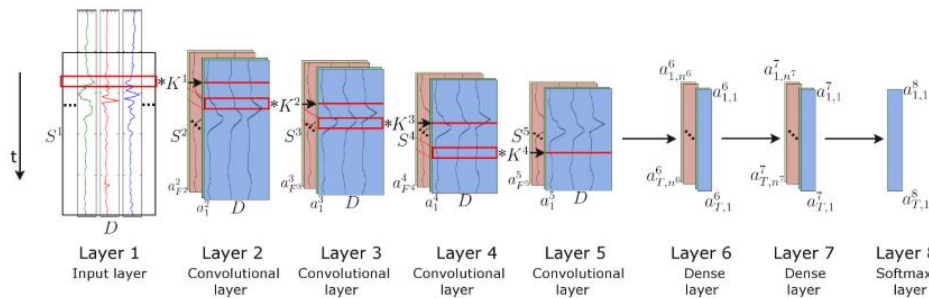
88   their model and similar architectures (e.g., Devineau et al., 2018, May) is that they do not capture the correlation

89   between variables as the feature extraction is carried out separately for each variable.

90

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

93   Brunel et al. (2019) present CNNs adapted for TSC in cosmology using 1D filters to extract features from each channel

94   over time and a 1D convolution in depth to capture the correlation between the channels. They compared the results

95 from LSTMs with CNNs, which shows that CNNs give better results than LSTMs. Nevertheless, both deep learning
96 approaches are very promising.
97 The combination of CNNs and LSTM units has already yielded state-of-the-art results in problems requiring
98 classification of temporal information such as human activity recognition (Li et al., 2017; Mutegeki and Han, 2020,
99 February), text classification (Luan and Lin, 2019; March, She and Zhang, 2018, December; Umer et al., 2020),
100 video classification (Lu et al., 2018 and Wu et al., 2015, October), sentiment analysis (Ombabi et al., 2020; Sosa,
101 2017; Wang et al., 2016, August; Wang et al., 2019), typhoon formation forecasting (Chen et al.,2019), and
102 arrhythmia diagnosis (Oh et al., 2018). In this architecture, convolutional operations capture features and LSTMs
103 capture time dependencies on extracted features. Ordóñez and Roggen (2016) propose a deep convolutional LSTM
104 model (DeepConvLSTM) for activity recognition (Fig. 2). Their results are compared to the results from standard
105 feedforward units showing that DeepConvLSTM reaches a higher F1 score and better decision boundaries for
106 classification. Furthermore, they noticed that the LSTM model gives promising results with relatively small datasets.
107 Furthermore, LSTMs present a better performance in capturing longer temporal dynamics, whereas the convolution
108 filters can only capture the temporal dependencies dynamics within the length of the filter.

109

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

111 This project is a part of a project called DAVE, which aims to develop a tool to provide regional ice jam watches and
112 warnings, based on the integration of three aspects: the current conditions of the ice cover; hydro-meteorological
113 patterns associated with breakup ice jams; and channel predisposition to ice-jam formation. The outputs of the previous
114 tasks will be used to develop an ice-jam monitoring and warning module and transfer the knowledge gained to end-
115 users to better manage the risk of ice jams.
116 The objective of this research is to develop deep learning models to predict breakup ice-jam events to be used as an
117 early warning system of possible flooding. While most TSC research in deep learning is performed on 1D channels
118 (Hatami et al., 2018, April), we propose deep learning frameworks for multivariate TSC for ice-jam prediction. The
119 objective of this research is to develop deep learning models to predict breakup ice-jam events to be used as an early
120 warning system of possible flooding. Through our comprehensive literature review, we noticed that CNN (e.g., Brunel
121 et al., Deep2019; Cui et al., 2016; Devineau et al., 2018, June; Kashiparekh, 2019, July; Nosratabadi et al., 2020;Yan
122 et al., 2020; Yang et al., 2015, June; Yi et al., 2017; Zheng et al., 2016), LSTM (e.g., Fischer and Krauss, 2018; Lipton
123 et al., 2015; Nosratabadi et al., 2020; Torres et al., 2021), and a combined CNN-LSTM (e.g., Karim et al., 2o17;

124 Livieris et al., 2020; Ordóñez and Roggen, 2016; Sainath et al., 2015, April; Xingjian et al., 2015) have been widely
125 used for TSC. There are numerous applications of CNN, LSTM, and their hybrid versions applied in hydrology
126 (Althoff et al., 2021; Apaydin et al., 2020; Barzegar et al., 2021, 2020; Kratzert et al., 2018; Wunsch et al., 2020;
127 Zhang et al., 2018). Although deep learning methods areseem to be promising to address the requirements of ice-jam
128 predictions., none of these methods yet have been explored for ice jam prediction.
129 Hence, we developed three deep learning models; a CNN, an LSTM, and a combined CN-LSTM (Convolutional
130 Long Short-Term Memory)CNN-LSTM for ice-jam predictions and compared the results. The previous studies show
131 that these models show good capabilities in capturing features and the correlation between features (through
132 convolution units) and time dependencies (through memory units) that will be later used for TSC. The previous studies
133 show that these models show good capabilities in capturing features and the correlation between features (through
134 convolution units) and time dependencies (through memory units) that will be later used for TSC. The combined
135 CNCNN-LSTM can reduce errors by compensating for the internal weaknesses of each model. In the CNCNN-LSTM
136 model, CNNs capture features, then the LSTMs give the time dependencies on the captured features.
137 Furthermore, we also developed some machine learning methods as simpler methods for ice-jam prediction. And their
138 results are compared with results from the developed deep learning models.

139 **2 MaterialMaterials and Methods**

140 **2.1 Input dataData and study area**

141 It is known that specific hydro-meteorological conditions lead to ice-jam occurrence (Turcotte and Morse, 2015,
142 August and White, 2003). For instance, breakup ice jams occur when a period of intense cold is followed by a rapid
143 peak discharge resulting from spring rainfall and snowmelt runoff (Massie et al., 2002). The period of intense cold
144 can be represented by the changes in Accumulated Freezing Degree Days (AFDD). The sudden spring runoff increase
145 is not often available at the jam location and can be represented by liquid precipitation and snow depth some days
146 before the ice-jam occurrence (Turcotte and Morse, 2015, August and White, 2003). For instance, breakup ice jams
147 occur when a period of intense cold is followed by a rapid peak discharge resulting from spring rainfall and snowmelt
148 runoff (Massie et al., 2002). The period of intense cold can be represented by the changes in Accumulated Freezing
149 Degree Days (AFDD). The sudden spring runoff increase is not often available at the jam location and can be
150 represented by liquid precipitation and snow depth some days before the ice-jam occurrence (Zhao et al., 2012).
151 Prowse and Bonsal (2004) and Prowse et al. (2007) evaluate various hydroclimatic explanations for river ice freeze-
152 up and breakup, concluding that shortwave radiation is the most critical factor influencing the mechanical strength of
153 ice and consequently the possibility of breakup ice jams to occur. Turcotte and Morse (2015, August) explain that
154 Accumulated Thawing Degree Day (ATDD), an indicator of warming periods, partially covers the effect of shortwave
155 radiation. In the previous studies of ice-jam and breakup predictions, discharge and changes in discharge, water level
156 and changes in water level, AFDD, ATDD, precipitation, solar radiation, heat budget, and snowmelt or snowpack are
157 the most readily used variables (Madaeni et al., 2020).
158 The inputs we used in this study are historical ice-jam or no ice-jam occurrence (Fig. 23) as well as hydro-
159 meteorological variables including liquid precipitation (mm), min and max temperature (°C), AFDD (from August

160 1st; °C), ATDD (from January 1st; °C), snow depth (cm) and net radiation (W m^{-2}) in ~~all~~150 rivers in Quebec. The net
161 solar radiation, the total energy available to influence the climate, is calculated as the difference between incoming
162 and outgoing energy. If the median temperature is greater than 1, the precipitation is considered liquid precipitation.
163 The statistics of hydro-meteorological data used in the models are presented in Table 1. The source, time period, and
164 spatial resolution of the input variables are ~~presented~~shown in Table ~~1. The "NaN" precipitation values get 0 values~~2.
165 ~~TThe ice~~Ice-jam database is provided by the Quebec Ministry of Public Security (MSPQ; Données Québec, 2021) for
166 150 rivers in Quebec, mainly in the St. Lawrence basin. The database comes from the digital or paper event reports
167 by local authorities under the jurisdiction of the MSPQ from 1985 to 2014. Moreover, some other data of this database
168 are provided by the field observations from the Vigilance / Flood application from 2013 to 2019. It contains 995
169 recorded jam events that are not validated and contain many inaccuracies, mainly in the toponymy of the rivers,
170 location, dating, and the redundancy of jam events.
171 The names of the watercourse of several ~~ice~~recorded jams are not given or completely wrong or affected by a typo or
172 an abbreviation. The toponymy of the rivers was corrected using the National Hydrographic Network (NHN; National
173 Hydrographic Network - Natural Resources Canada (NRCan)), the Geobase of the Quebec hydrographic network
174 (National Hydro Network - NHN - GeoBase Series - Natural Resources Canada), and the Toporama Web map service
175 (The Atlas of Canada - Toporama - Natural Resources Canada) of the Sector of Earth Sciences.
176 Several ice jams are placed on the banks at a small distance (less than 20 m) from the polygon of the river. In this
177 case, the location of the ice jam is moved inside the river polygon. In other cases, ~~the~~ ice-jam point is posed further
178 on the flooded shore at a distance between 20 and 200 m. This has been corrected based on images with very high
179 spatial resolution, the sinuosity and the narrowing of the river, the history of ice jams at the site in question, and the
180 press archives. In addition, some ice jams were placed too far from the mentioned river due to ~~a typo in entering~~
181 ~~their~~wrong recorded coordinates in the database. A single-digit correction in longitude or latitude returned the jam to
182 its exact location. There are certain cases where the date of jam formation is verified by searching the press archives,
183 notably when the date of formation is missing or several jams with the same dates and close locations in a section of
184 a river are present.
185 The ice jam database contains many duplicates. This redundancy can be due to merging two data sources, the double
186 entry during ice-jam monitoring, or recording an ice jam for several days. The duplicates are removed from the
187 database. The corrected ice-jam database contains 850 jams for 150 rivers, mainly in southern Quebec (Fig. 3). The
188 ice jams formed in November and December (freeze-up jams) are removed to only include breakup jams (from January
189 15th) in the modelling as these two types of jams are formed due to different processes. The final breakup ice-jam
190 database that used in this study includes 504 jam events.

191

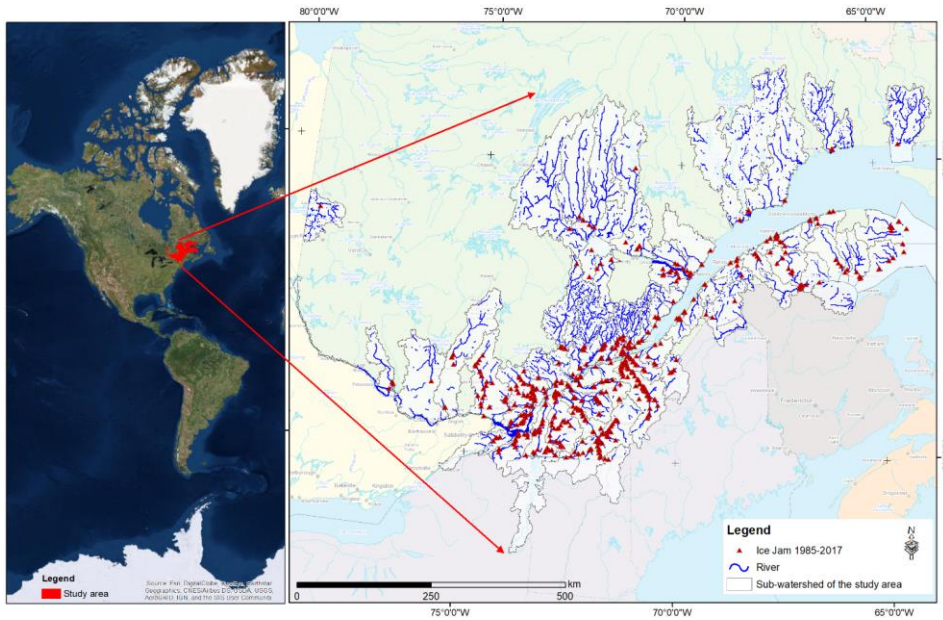



Figure 3. Study area and historic ice-jam locations recorded in Quebec from 1985-2017.

Table 1. ~~Hydro~~Statistics of hydro-meteorological ~~data~~variables used ~~as~~in the ~~input to the model~~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

194
195

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

196 * The average was used to derive the AFDD and the ATDD.
197

2.2 Machine learning models for TSC

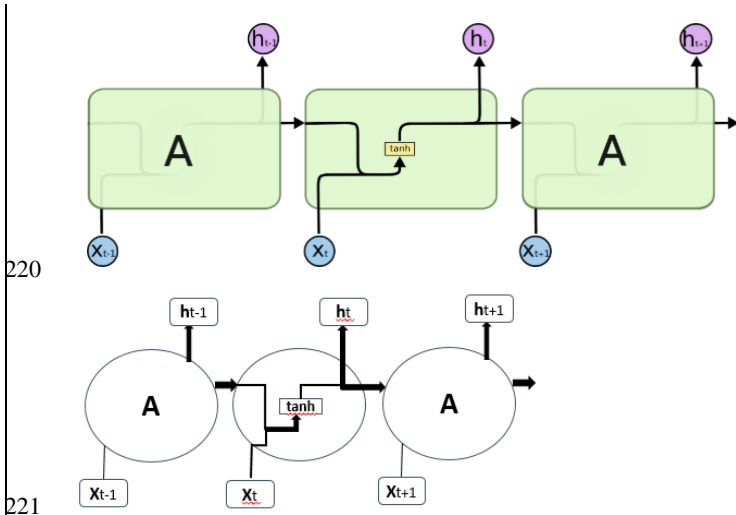
The common machine learning techniques that have been used for TSC are SVM (Rodríguez and Alonso, 2004; Xing and Keogh, 2010), KNN (Li et al., 2013; Xing and Keogh, 2010), decision tree (DT; Brunello et al., 2019; Jović et al., 2012, August), and multilayer perceptron (MLP; del Campo et al., 2021; Nanopoulos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

We developed the mentioned machine learning methods and compared their results with the results 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' solver (which can converge faster and perform better for small datasets), alpha of 1e-5, and 3 layers with 7 neurons in each layer.

2.3 Deep learning models for time-series classification (TSC)TSC

The most common and popular deep neural networks for TSC are MLPMLPs, CNNs, and LSTM.LSTMs (Brownlee, 2018; and Torres et al., 2021). Despite their power, however, MLP has limitations 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). In this regard, Recurrent Neural Networks (RNNs) are specifically adapted to sequence data through the direct connections between individual layers (Jozefowicz et al., 2015). Recurrent Neural Networks perform the same repeating function with a straightforward structure, e.g., a single tanh (hyperbolic tangent) layer, for every input of data (x_t), while all the inputs are related to each other with their hidden internal state, which allows it to learn the temporal dynamics of sequential data (Fig. 4).

220

221

222
223
Figure 4. An RNN with a single tanh layer, where A is a chunk of the neural network, ~~xt~~x is input data, and ~~ht~~h is output data ~~(after Olah, 2015).~~.

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

228 Long short-term memory RNNs are developed to improve the performance of RNNs by integrating a memory to
229 model long-term dependencies in time-series problems (Brunel et al., 2019; Karim et al., 2019). ~~-~~Long short-term
230 memory networks do not have the problem of exploding gradients. The LSTMs have four interacting neural network
231 layers in a very special way (Fig. 5). An LSTM has three ~~gates (~~sigmoid (σ) layers~~; σ)~~ to control how much of each
232 component should be let through by outputting numbers between zero and one. The input to an LSTM goes through
233 three gates ("forget", "input", and "output gates") that control the operation performed on each LSTM block (Ordóñez
234 and Roggen, 2016). The first step is the "forget gate" layer that gets the output of the previous block (h_{t-1}), the input
235 for the current block (X_t), and the memory of the previous block (C_{t-1}) and gives a number between 0 and 1 for each
236 number in the cell state (C_{t-1}; Olah, 2015). The second step is called the "input gate" with two parts, a sigmoid layer
237 that decides which values to be updated and a tanh layer that creates new candidate values for the cell state. These two
238 new and old memories will then be combined and control how much the new memory should influence the old
239 memory. The last step (output gate~~; step 3 in Fig. 5~~) gives the output by applying a sigmoid layer deciding how much
240 new cell memory goes to output, and multiply it by tanh applied to the cell state (giving values between −1 and 1).

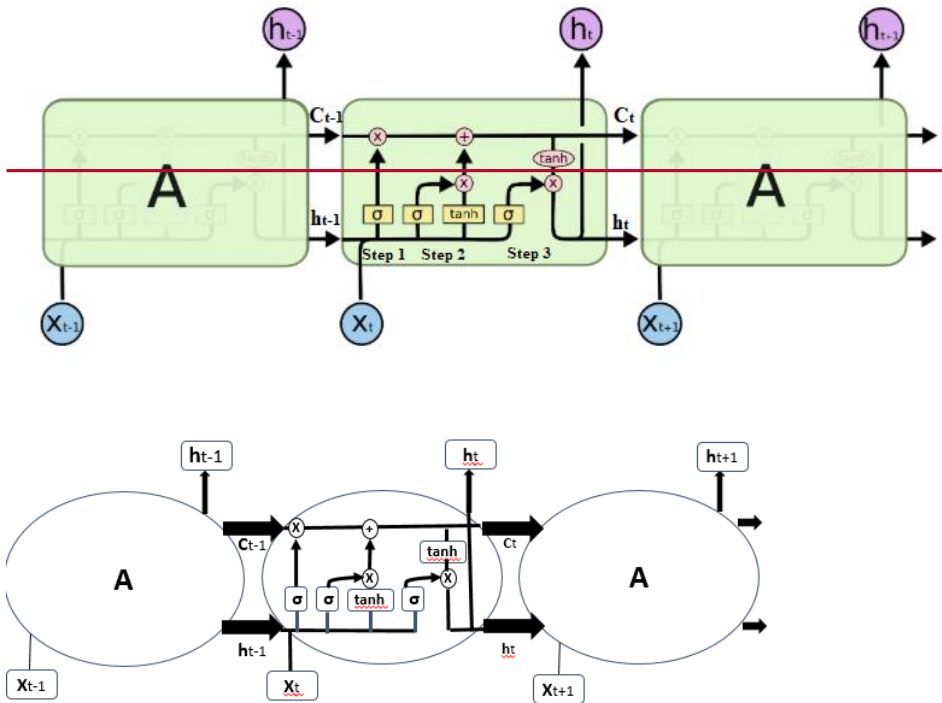



Figure 5. Structure of LSTM block with four interacting layers ~~(adopted from Olah, 2015).~~.

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

~~Convolutional neural networks~~The CNNs are the most widely used deep learning methods in TSC problems (Fawaz et al., 2019). They learn spatial features from raw input time series using filters (Fawaz et al., 2019). ~~Convolutional neural networks~~The CNNs are robust and need a relatively small amount of training time comparing with 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 ~~layer~~ (or filter~~).~~) layer. A CNN usually involves several convolutional layers, activation functions, and pooling layers for feature extraction following by dense layers (or MLP) as a classifier (Devineau et al., 2018, June). 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 neurons to all neurons in the previous layer. Therefore, each neuron in CNNs is connected to only a local region of

258 the input, namely the receptive field, which equals the filter size (Fig. 5̶6). This feature reduces the number of
259 parameters by limiting the number of connections between neurons in different layers. The input is first convolved
260 with a learned filter, and then an element-wise nonlinear activation function is applied to the convolved results (Gu et
261 al., 2018). The pooling layer performs a downsampling operation such as maximum or average, reducing the spatial
262 dimension (Fig. 6).. One of the most powerful features of CNNs is called weight or parameter sharing, where all
263 neurons share filters (weights) in a particular feature map (Fawaz et al., 2019) to reduce the number of parameters.

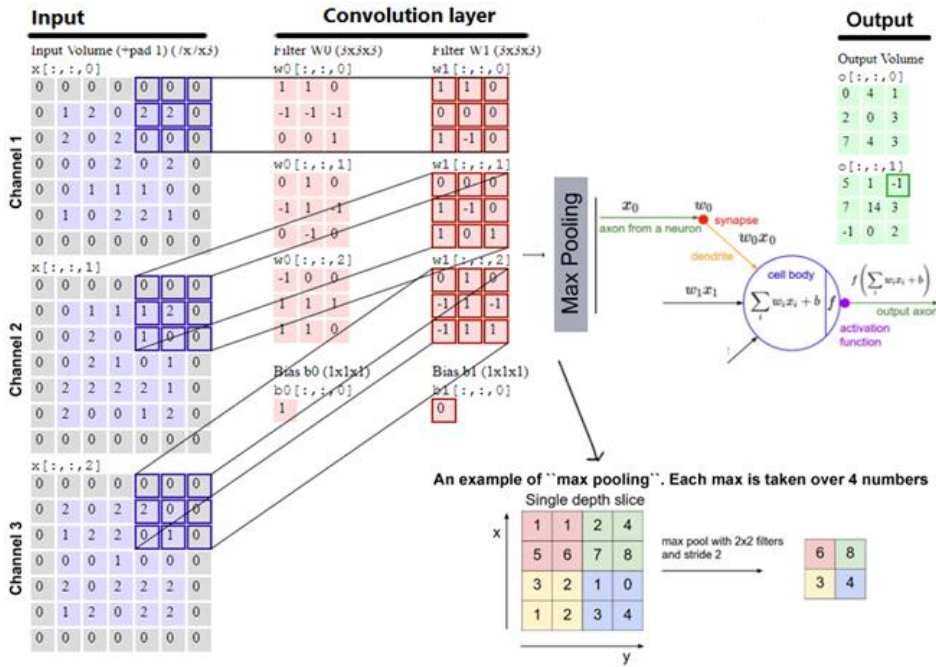



Figure 6. . A CNN Architecture for image classification (modified from Karpathy, 2017).

264

265 **Figure 6. A convolution layer structure including two sets of filters.**

266

267 **2.~~3~~4 Model libraries**

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

274 **2.~~4~~5 Preprocessing**

275 The data is comprised of variables with varying scales, and the machine learning algorithms can benefit from rescaling
276 the variables to all have the same scale. Scikit-learn (Pedregosa et al., 2011) is a free library for machine learning in
277 Python that can be used to preprocess data. We examined Scikit-learn MinMaxScaler (scaling each variable between
278 0 and 1), Normalizer (scaling individual samples to the unit norm), and StandardScaler (transforming to zero mean
279 and unit variance separately for each feature). The results show that MinMaxScaler (Eq. (1)) ~~works~~leads to the ~~best in~~
280 ~~our models~~most accurate results. The scaling of validation data is done with min and max from train data.

281 $$X_{scaled} = \left(\frac{X - X.min}{X.max - X.min} \right), \frac{X - X.min}{X.max - X.min} \qquad (1)$$

282 For each jam or no jam event, we used 15 days of information before the event to predict the event on the 16th day.
283 We generate a balanced dataset with the same number of jam and no-jam events (1008 small sequences totally),
284 preventing the model from becoming biased to jam or no-jam events. The hydro-meteorological data related to no-
285 jam events are constructed by extracting data from the reaches of no-jam records. To examine models' generalization,
286 we hold out 10% of data for testing and 80 % and 20 % of remaining data for training and validation, respectively.
287 We used ShuffleSplit subroutine from the Scikit-learn library, where the database was randomly sampled during each
288 re-shuffling and splitting iteration to generate training and validation sets. We applied 100 re-shuffling and splitting
289 iterations ~~with 80 % of data~~ for training and ~~20 % for~~ validation. There are ~~806~~726, 181, and ~~202~~101 small sequences
290 with the size of (16, 7), 16 days of data for the seven variables; for training ~~and,~~ validation, and test, respectively. ~~To~~
291 ~~examine models' generalization, we hold out 30 small sequences for testing and 80 % and 20 % of remaining data for~~
292 ~~training and validation, respectively.~~

293 **2.~~5~~6 Training**

294 Training a deep neural network with an excellent generalization to new unseen inputs is challenging. As a benchmark,
295 a CNN model with the parameters and layers similar to previous studies (e.g., Ordóñez and Roggen, 2016) is
296 developed. The model shows underfitting or overfitting with various architectures and parameters. To overcome
297 underfitting, deeper models and more nodes in each layer are beneficial; however, overfitting is more challenging to
298 overcome. ~~The ice~~Ice-jam dataset for Quebec contains 1008 balanced sequence instances (with a length of 16), which
299 is small~~, which easily causes the network~~ for deep learning. The deep learning models often tend to ~~memorize~~overfit

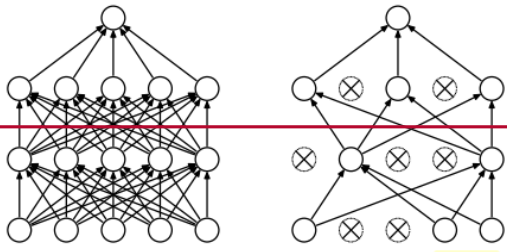
300 small datasets by memorizing inputs rather than training ~~examples and consequently results in overfitting~~, as a small
301 dataset may not appropriately describe the relationship between input and output spaces.

2.~~5~~6.1 Overcome overfitting

There are various methods to tackle 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. **2.5.1.1 Noise layer**

308 ~~The first approach to overcome overfitting is acquiring more data that is not possible with ice-jam records. Another~~
309 ~~popular approach to increase the number of samples is data augmentation, including cropping, rotating, blurring, color~~
310 ~~modification, and noise injection in image classification. Data augmentation can act as a regularizer, prevent~~
311 ~~overfitting, and improve performance in imbalanced class problems (Wong et al., 2016). However, the application of~~
312 ~~data augmentation in deep learning for time series classification still has not been studied thoroughly (Fawaz et al.,~~
313 ~~2019). To expand the size of the dataset, noise layers, as a simple form of random data augmentation, can be used.~~
314 ~~Over the training process, each time an input sample is exposed to the model, the noise layer creates new samples in~~
315 ~~the vicinity of the training samples resulting in various input data every time, increases randomness, making the model~~
316 ~~less prone to memorize training samples and learns more general features (resulting in better generalization).~~
317 ~~We added the Gaussian noise layer (from the Lasagne library), where the noise values are Gaussian-distributed with~~
318 ~~zero-mean and a standard deviation of 0.1 to the input.~~ ~~The noise layer is usually added to the input data but can also~~
319 ~~be added to other layers.~~

~~2.5.1.2 Dropout~~

321 ~~The other approach to tackle overfitting is dropout (Srivastava et al., 2014). The dropout, the most successful method~~
322 ~~for neural network regularization, randomly sets inputs to zero (Fig. 7). To overcome overfitting and examine the~~
323 ~~effectiveness of dropout in our models, the dropout with the recommended rates of 0.1 for the input layer and between~~
324 ~~0.5 and 0.8 for hidden layers (Garbin et al., 2020) are applied in different layers of the models.~~

326 ~~Figure 7. A neural network with two hidden layers (left) and a neural network with dropout (right; after Srivastava et al.,~~
327 ~~2014).~~

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 a noise layer (Fig. 7). Adding a noise layer to other layers does not improve any of the developed models for ice-jam prediction.

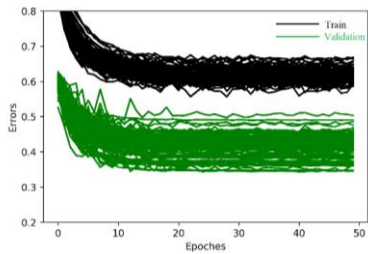
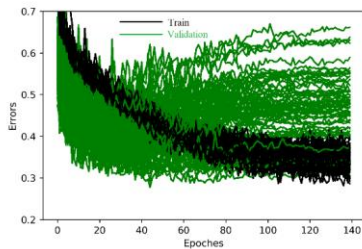



Figure 7. Train and validation errors over epochs for CNN-LSTM model with a noise layer.

Early stopping is another efficient method ~~to tackle overfitting via halting~~that halts the training procedure where further training would decrease training loss, while validation loss starts to increase.

~~As explained earlier, the input data is scaled separately for each feature to be between 0 and 1. However, in deep learning, the distribution of the input of each layer will be changed by updates to all the preceding layers, so-called internal covariate shift. Hence, hidden layers try to learn to adapt to the new distribution slowing down the training process. Batch normalization (Ioffe and Szegedy, 2015, June) is a recent method that provides any layer with inputs of zero mean and unit variance and consequently prevents internal covariate, solves exploding or vanishing gradient problems, allows the use of higher learning rates, improves the training efficiency, and speeds up the training. Batch normalization adjusts the value for each batch, results in more noise acting as a regularizer, similar to dropout, and thus reduces the need for dropout (Garbin et al., 2020). We performed~~Neural networks solve an optimization problem that requires a loss function to calculate the model error. 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 (Fig. 8). More explanations about the other methods that are used in this study to overcome overfitting (e.g., batch normalization ~~over each channel in different layers in our models to find its best locations through trial and error.~~, and L2 regularization) can be found in the Appendix.

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

352 **2.5.1.5 Regularization**

353 There are two general ways to keep a deep neural network simple and consequently prevent overfitting; through the
354 number of weights and values of weights. The number of weights can be controlled by the number of layers and nodes
355 optimized via the grid or random search. A network with large weights can be more complex and unstable as large
356 weights increase loss gradients exponentially, resulting in exploding gradients that cause massive output changes with
357 minor changes in the inputs. In turn, the exploding gradients can force the model loss and weights to "NaN" values
358 (Brownlee, 2017).

359 The simplest and most common approach to keep the weights small is regularization methods that involve checking
360 model weights and adding an extra penalty term to the loss function in proportion to the size of weights' size in the
361 model. The two main methods used to calculate the size of the weights are L1 (the sum of the absolute values of the
362 weights; Eq. (2)) and L2 or weight decay (the sum of the squared values of the weights; Eq. 3). In Eq. (2) and (3), λ
363 is a parameter that controls the importance of the regularization, and W is the network parameters. The L1
364 regularization encourages weights to be 0.0 (causing underfitting) and very few features with non-zero weights, while
365 L2 regularization forces the weights to be small rather than zero. Hence, L2 can predict more complex patterns when
366 output is a function of all input features. We used an L2 regularization cost by applying a penalty to the parameters of
367 all layers in the networks in CNN, LSTM, and CN-LSTM models.

368 $$Cost\ function + \lambda \sum_{i=1}^{n} |w_i| \tag{2}$$

369 $$Cost\ function + \lambda \sum_{i=1}^{n} w_i^2 \tag{3}$$

370 **2.5.2 Architecture Tuning**

371 **2.6.2 Model Hyperparameters**

372 Finding hyperparameter values in deep learning has been challenging due to the complex architecture of deep learning
373 models and a large number of parameters (Garbin et al., 2020). To find the best model architecture, we study the
374 performance of models with different layers and parameters such as number of noise, batch normalization,
375 convolutional, pooling, LSTM, dropout, and dense layers, as well as different pooling sizes and strides, different batch
376 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 are optimized through manual trial and error searches as grid search experiments suffer from poor coverage in dimensions (Bergstra and Bengio, 2012) and manual experiments are much easier and more interpretable in 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 information about other parameters readers are referred to the Appendix.

2.5.2.1 Activation function

The activation function adds non-linearity to the network allowing the model to learn more complex relationships between inputs and outputs (Zheng et al., 2014, June). Each activation function that is used in deep learning has its advantages and disadvantages, and typical activation functions in deep learning are Rectified Linear Unit (ReLU; Eq. (4)), sigmoid (Eq. (5)), and hyperbolic tangent (tanh; Eq. (6); Fig. 8; Gu et al., 2018). In deep neural networks, adding more layers with certain activation functions results in the vanishing gradient problem where the gradients of the loss function become almost zero, causing difficulties in training. For instance, the sigmoid function maps a large input space into a small one between 0 and 1. Hence, when the input is very positive or very negative, the sigmoid function saturates (becomes very flat) and becomes insensitive to small changes in its input, causing the derivatives to disappear (Goodfellow et al., 2016). Therefore, in backpropagation, small derivatives are multiplied together, causing the gradient to decrease exponentially, propagating back to the first layer. This causes ineffective updates of weights and biases of the initial layers and consequently inaccuracy. Some solutions to overcome this problem include using specific activation functions like ReLU and tanh and using batch normalization layers to prevent the activation functions from becoming saturated. The ReLU recently drown lots of attention and has been widely used in recent deep learning models (Gamboa, 2017). The advantage of ReLU over sigmoid and tanh is a better generalization, making the training faster and simpler. Hence, we investigated the performance of the model with ReLU, sigmoid, or tanh activation functions in convolutional layers.

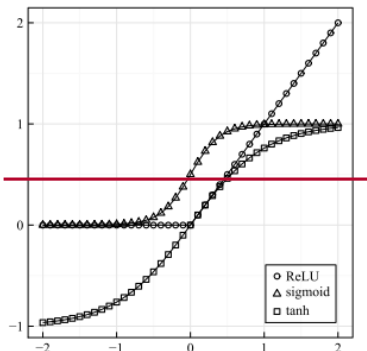
$$ReLU(x) = max\ (0, x) \tag{4}$$

$$Sigmoid(x) = \frac{1}{1+e^{-x}}$$

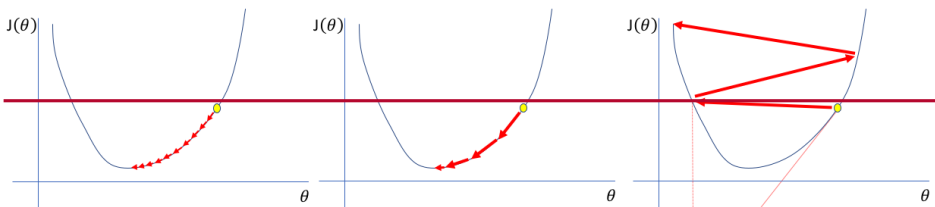
Table 3. Common values and selected values for different parameters of the models.

Parameter	Common values	Selected value
Mini-batch size	16, 32, 64	16
Number of convolution filters	32, 64, 128	128
Filter size	3, 5, 7	(5,1) and (5,3)
Number of LSTM units	32, 64, 128	128
Number of dense layer units	16, 32, 128, 256	32
Momentum in SGD	0.5, 0.99, 0.9	0.9

2.6.2.1 (5)

406 $$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{6}$$

407
408 **Figure 8. Illustration of sigmoid, tanh, and ReLU activation functions (after Zheng et al., 2016).**

409 **2.5.2.2 Learning rate**

410 To find the minimum cost function, a move in the negative direction of the gradient is required. This movement is
411 called the "learning rate," which is the most significant hyperparameter in training a deep neural network. The model
412 error is calculated, and the errors corresponding to weights updated by the learning rate are backpropagated in the
413 network. A too-small learning rate needs many updates and epochs, reaching the minimum. On the other hand, a too-
414 large learning rate causes dramatic updates and leads to oscillations in loss over epochs. A good learning rate quickly
415 reaches the minimum point between 0.1 to 1e-6 on a log scale and can be found through a grid or random search (Fig.
416 9).

417
418 **Figure 9. Too small, good, and too large learning rates from left to right (after Jordan, 2018).**

419 **2.5.2.3 Update expression**

420 There are various algorithms to update the trainable parameters at each mini-batch. The parameter updating procedure
421 includes feedforwarding, backpropagation, and applying gradients. We tried the Stochastic Gradient Descent (SGD)
422 with Nesterov momentum, RMSProp, Adadelta, and Adam updates to update the parameters in Lasagne. The SGD
423 with momentum updates the model weights by adding a momentum so that the overall gradient depends on the current
424 and previous gradients, causing the weights to move in the previous direction without oscillation.

2.5.3 Network optimization

Training CNN involves global optimization by defining a loss expression to be minimized overtraining. For the classification task, the loss function of the models is calculated using categorical cross-entropy between network outputs and targets (Eq. (7)), where L is the loss, p is the prediction (probability), t is the target, and c is the number of classes. Then, the mean of the loss is computed over each mini-batch.

$$L = -\sum_{i=1}^{c=2} t_i \log(p_i) \tag{7}$$

2.5.4 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 2. Therefore, the F1 score (Eq. (8)), which considers each class equally important, is used to measure the binary classification accuracy. The F1 score, as a weighted average of the precision (Eq. (9)) and recall (Eq. (10)), has the best and worst scores of 1 and 0, respectively. In Eqs. 9 and 10, TP, FP, and FN are true positive, false positive, and false negative, respectively.

Table 2. The number of jam and no jam events in train and validation and test datasets.

=	Train and validation	Test
Jam	504	48
No jam	403	53

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{8}$$

$$Precision = \frac{TP}{TP+FP} \tag{9}$$

$$Recall = \frac{TP}{TP+FN} \tag{10}$$

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 network generalization using a test dataset, which is not seen during training and validation.

3.1 Hyperparameters optimization

3.1.1 Batch size

The inputs and corresponding targets are iterated in mini-batches for training and validation. Batch size significantly influences the training time (Fawaz et al., 2019, July), and the batch size of 32 is usually used in previous studies. However, we investigated batch sizes of 16, 32, and 64, and the mini-batches of 16 demonstrate to improve the results slightly.

3.1.2 Noise layers

The performance of CNN and LSTM models developed for the ice-jam prediction problem is improved by adding a noise layer to the input, while the CN-LSTM model showed underfitting. Adding a noise layer to other layers does not improve any of the developed models for ice-jam prediction.

3.1.3 Dropout layer

Adding dropout layers could not improve any developed models. This agrees with previous studies revealing that dropout does not work well with LSTMs (Zaremba et al., 2014) and CNNs, and dropout layers do not work when batch size is small (less than 256; Garbin et al., 2020). Furthermore, it is in agreement with Garbin et al. (2020) stating that utilizing batch normalization layers in a model reduces the need for dropout layers.

3.1.4 **Number of layers**

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

3.1.52.6.2.2 Number and size of CNconvolution filters

Fawaz et al. (2019, July) explain the number and length of filters used in CNNs. Data with more classes need more filters to classify the inputs accurately. Longerand longer time series need longer filters to capture longer patterns and consequently to produce accurate results. (Fawaz et al., 2019, July). However, longer kernelsfilters significantly increase the number of parameters and increase the potential for overfitting small datasets, while a small kernelfilter size risks poor performance. In our models, the optimum number of filters is attained to be 128 by searching among the typical number of filters (i.e., 32, 64, and 128). The kernel sizes of 3, 5, and 7 are often applied in deep CNNs. We tried these filter sizes, and the best performance was achieved through usingfinally selected two convolutional layers with 1-D filters of (5, 1) with theand stride of (1, 1) to capture temporal variation for each variable separately.

483 Furthermore, one convolutional layer with 2-D filters of size (5, 3) ~~with the~~and stride of (1, 1) is then used to
484 ~~achieve~~capture the correlation between variables via depth-wise convolution of input time-series. A big stride might
485 cause the model to miss valuable data used in predicting and smoothing out the noise in the time series. The layers in
486 CNNs have a bias for each channel, sharing across all positions in each channel.

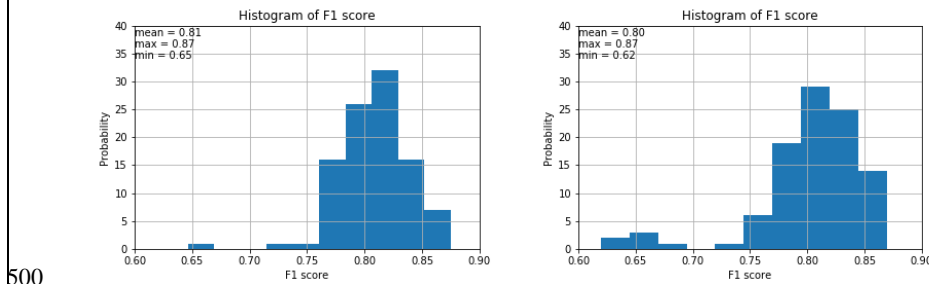
487 ### ~~3.1~~2.6 Padding

488 ~~The convolution is applied where the input and the filter overlap. Hence, we pad the input by zeros with half the filter~~
489 ~~size on both sides. Using stride of 1 with "Pads = same" (in Lasagne) in the convolutional 2-D layers results in an~~
490 ~~output size equal to the input size for each layer.~~

491 ### ~~3.1.7 Activation functions in CN layers~~

492 ~~The experiments demonstrate that errors are very high using tanh, whereas ReLU and sigmoid show almost the same~~
493 ~~performance. As ReLU performs slightly better than sigmoid, we used ReLU in our models.~~

494 ### ~~3.1.8 Weight initialization~~

495 ~~Among the various types of methods available in Lasagne for weight initialization, the GLOROT uniform (i.e., Xavier;~~
496 ~~Glorot and Bengio, 2010, March) and He initializations (He et al., 2015), the most popular initialization techniques,~~
497 ~~are used to set the initial random weights in convolutional layers. The results reveal that these methods yield almost~~
498 ~~the same F1 scores. However, the histograms of F1 scores reveal that GLOROT uniform yields slightly better results~~
499 ~~(Fig. 10).~~

501
502 ~~Figure 10. Histograms of F1 score for CNN using He (left) and GLOROT uniform (right) weight initialization with 100 random train-validation splits.~~

503 ### ~~3.1.9 Number of LSTM units and their activation functions~~

504 ~~The optimal number of units in LSTM layers was found through a search over typical numbers of 32, 64, and 128.~~
505 ~~We found that 128 units yield the best results in our models. We used the default activation function of tanh in LSTM~~
506 ~~layers.~~

507 ~~3.1.10 Dense layer~~

508 ~~The dense layers with RecLU functions following by one dense layer with softmax function are applied after the~~
509 ~~feature learning and LSTM layers to perform classification. The common number of units in dense layers are 16, 32,~~
510 ~~128, and 256. We found that 32 gives the best results in our models. To output the binary classes from the network,~~
511 ~~softmax or sigmoid functions can be used. We applied softmax as it gives a probability for each class where their total~~
512 ~~sum is one.~~

513 ### ~~3.1.11.~~2.4 Adaptive learning rates

514 The adaptive learning rate decreases the learning rate and consequently weights over each epoch. We tried different
515 base learning and decay rates for each model and found that the learning rate significantly impacts the model
516 performance. Finally, we chose a base learning rate of 0.1, 0.01, and 0.001 for LSTM, CNN, and ~~CN~~CNN-LSTM ~~and~~,
517 respectively. A decay rate of 0.8 was used for CNN and ~~CN~~CNN-LSTM, while for the LSTM model, this rate was
518 0.95. Table ~~3~~4 shows the adaptive learning rates for CNN, LSTM, and ~~CN~~CNN-LSTM calculated using Eq. (~~11~~2) for
519 each epoch.

520 $\text{adaptive learning rate} = base\ learning\ rate\ \times\ decay^{epoch}$ ~~Eq. (11~~
521 (2)

522 The experiments show that the learning rate is the most critical parameter influencing the model performance. A small
523 learning rate can cause the ~~cost~~loss function to get stuck in local minima, and a large learning rate can result in
524 oscillations around global minima without reaching it.
525 Our ~~CN~~CNN-LSTM model is deeper than the other two models, and deeper models are more prone to a vanishing
526 gradient problem. To overcome the vanishing gradients, it is recommended that lower learning rates, e.g., lower than
527 1e-4, be used. Interestingly, we found that our ~~CN~~CNN-LSTM model works better with lower learning rates than the
528 other two models.
529

530 **Table ~~3~~4. The adaptive learning rate for 50 epochs.**

Epochs	Learning rate		
	CNN	~~**CN**~~**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	-

531

532

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 and worst scores of 1 and 0, respectively. In Eqs. 7 and 8, TP, FP, and FN are true positive, false positive, and false negative, respectively.

Table 5. The number of jam and no jam events in train and validation and test datasets.

	Train and validation	Test
Jam	456	48
No jam	451	53

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3}$$

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

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 network generalization using a test dataset, which is not seen during training and validation.

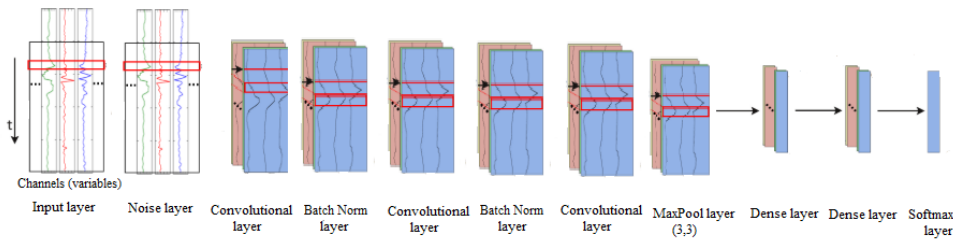
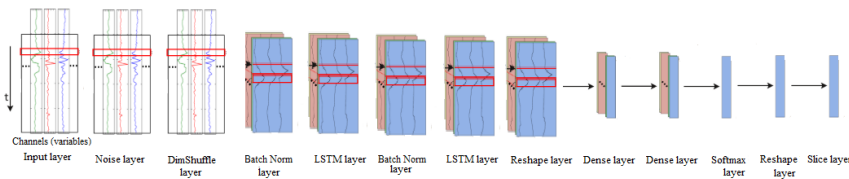
3.1.12 Update expression

We found that SGD with momentum works better than other methods in our cases. The typical values for momentum are 0.99, 0.9, and 0.5. We applied different values and found that 0.9 gives the best results in our models; this high momentum results in larger update steps. It is recommended to scale the learning rate by "1 – momentum" for using the high momentums, which gives 0.1. Interestingly, we already have applied the base learning rate of 0.1 for the LSTM model chosen through trial and error (as explained earlier); however, smaller values are chosen for CNN and CN-LSTM networks.

561 **3.22.7 Architecture of models**

562 The architectures of CNN, LSTM, and ~~CN~~CNN-LSTM models that are finally selected are presented in Figs. ~~11, 12~~9,
563 10, and ~~13~~11, respectively. The layers, their output shapes, and their number of parameters are presented in Tables ~~4,~~
564 ~~5~~6, 7, and ~~6~~8 for CNN, LSTM, and ~~CN~~CNN-LSTM models, respectively.
565 ~~The ice-jam dataset for Quebec contains 1008 balanced sequence instances (with a length of 16), which is small for~~
566 ~~deep learning. The deep learning models often tend to overfit small datasets by memorizing inputs rather than training.~~
567 ~~The noise layers applied to the CNN and LSTM models significantly overcome the overfitting problem through data~~
568 ~~augmentation. However, the performance of the CN-LSTM model dramatically deteriorates, including a noise layer~~
569 ~~(Fig. 14; showing underfitting).~~
570 The CNN models often include pooling layers to reduce data complexity and dimensionality. However, it is not always
571 necessary that every convolutional layer is followed by a pooling layer in the time-series domain (Ordóñez and
572 Roggen, 2016). For instance, Fawaz et al. (2019, July) do not apply any pooling layers in their models for TSC. We
573 tried max-pooling layers after different convolutional layers in CNN and ~~CN~~CNN-LSTM networks and found that a
574 pooling layer following only the last convolutional layer improves the performance of both models. This can be due
575 to subsampling the time series and using time series with a length of 16 that reduces the need for reducing
576 dimensionality.

577

578

579 **Figure ~~11~~9. The architecture of the CNN model for ice-jam prediction (adapted after Ordóñez and Roggen, 2016).**

580
581 **Figure ~~12~~10. The architecture of the LSTM model for ice-jam prediction (adapted after Ordóñez and Roggen, 2016).**

582

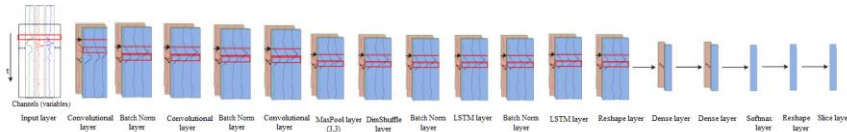



Figure ~~13~~11. The architecture of the ~~CN~~CNN-LSTM model for ice-jam prediction (adapted after Ordóñez and Roggen, 2016).

Table ~~4~~6. The layers, their 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

586

Table ~~5~~7. The layers, their 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

588

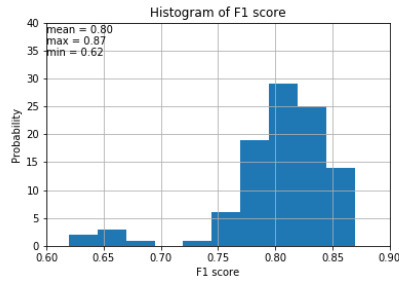
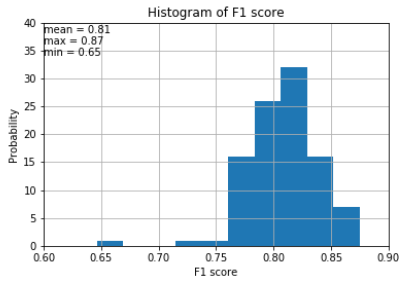
Table ~~6~~8. The layers, their output shapes, and their number of parameters for the ~~CN~~CNN-LSTM model.

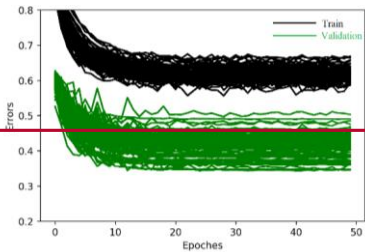
Layers	Output shape	Number of parameters
Input	(16, 1, 16, 7)	0
Conv2D	(16, 128, 16, 7)	640
BatchNorm	(16, 128, 16, 7)	512
Nonlinearity	(16, 128, 16, 7)	0
Conv2D	(16, 128, 16, 7)	81920
BatchNorm	(16, 128, 16, 7)	512
Nonlinearity	(16, 128, 16, 7)	0
Conv2D	(16, 128, 16, 7)	245888
MaxPool2D	(16, 128, 5, 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

590

3 Results and Discussion

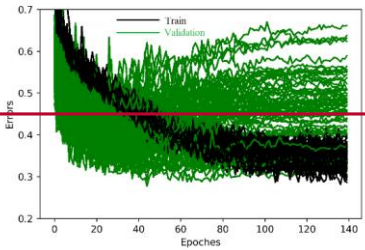
3.1 Weight initialization

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

598

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

601

602 ~~Figure 14. Train and validation errors over epochs for CN-LSTM model with a noise layer.~~

603 ~~3.~~3.2 **Model evaluation**

604 ~~LSTM needs only early stopping at 40 epoch among the developed models, as its validation error starts to increase,~~
605 ~~while its training error continues to decrease (Fig. 15). Hence, we set the number of epochs to 40 for the LSTM model.~~

606

607 ~~Figure 15. Train and validation errors over epochs for an LSTM model showing overfitting after 40 epochs.~~

608 ~~3.~~3.2.1 **Learning curves and F1 scores**

609 Line plots of the loss (i.e., learning curves), which are loss over each epoch, are widely used to examine the
610 performance of models in machine learning. Furthermore, line plots clearly indicate common learning problems, such

611 as underfitting or overfitting. The learning curves for CNN, LSTM, and ~~CN~~CNN-LSTM models are presented in Fig.

612 ~~16~~13. The LSTM model starts to overfit at epoch 40, so an early stopping is conducted. ~~CN~~CNN-LSTM performs

613 better than the other two models, as its training loss is the lowest and is lower than its validation loss. Histograms of

614 F1 scores (Fig. ~~16~~14 and Table ~~7~~9) show that ~~CN~~CNN-LSTM outperforms the other two models since it results in the

615 highest average and the ~~lowest~~highest minimum F1-scores for validation (0.82 and 0.75, respectively). Figure ~~16~~13

616 shows that the training error of CNN is lower than that of LSTM, which means that CNN trained better than LSTM

617 model. However, it is not true for the validation error. The reason that the validation error is less than the training error

618 in the LSTM model can be the employment of regularization methods as LSTM models are often harder to regularize,

619 agreeing with previous studies (e.g., Devineau et al., 2018, June).

620 The LSTM network is ~~valdated~~validated better than the CNN model since its average and minimum F1 scores for

621 validation are better than the CNN model (by 1 % and 32 %, respectively), and also LSTM yielded no F1 scores below

622 0.74 (Fig. ~~17~~14 and Table ~~7~~)~~. This reveals that LSTM is showing underfitting.~~9).

623 As shown in Fig. ~~16~~13, training loss is higher than validation loss in some of the results. ~~Some~~There are some reasons

624 ~~are~~ explaining that. Regularization reduces the validation ~~and testing (i.e., evaluation)~~ loss at the expense of increasing

625 training loss. The regularization techniques such as noise layers are only applied during training, but not during

626 ~~evaluation~~validation resulting in more smooth and usually better functions in ~~evaluation~~validation. There is no noise

627 layer in ~~CN~~CNN-LSTM model that may ~~caused~~cause a lower training error than validation error. However, other

628 regularization methods such as L2 regularization are used in all the models, including the ~~CN~~CNN-LSTM model.

629 Furthermore, the other issue is that batch normalization uses the mean and variance of each batch in training, whereas,

630 in ~~evaluation~~validation, it uses the mean and variance of the whole training dataset. Plus, training loss is averaged

631 over each epoch, while ~~evaluation~~validation losses are calculated after each epoch once the current training epoch is

632 completed. Hence, the training loss includes error calculations with fewer updates.

633 Among the developed machine learning models, SVM shows the best validation performance (Figure 15 and

634

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).

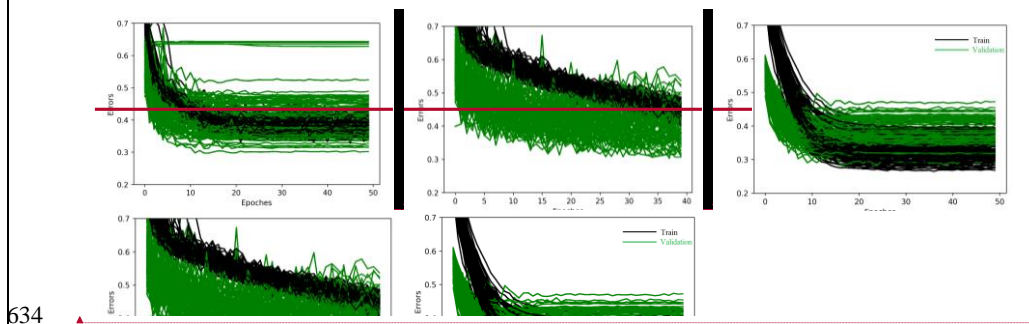
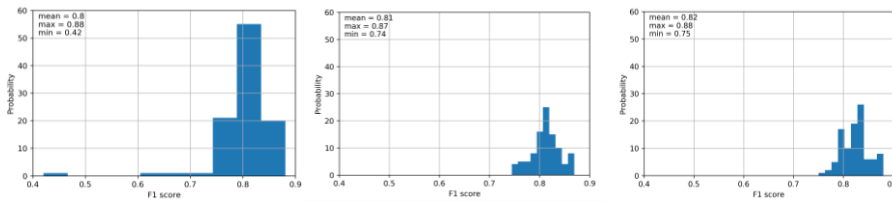
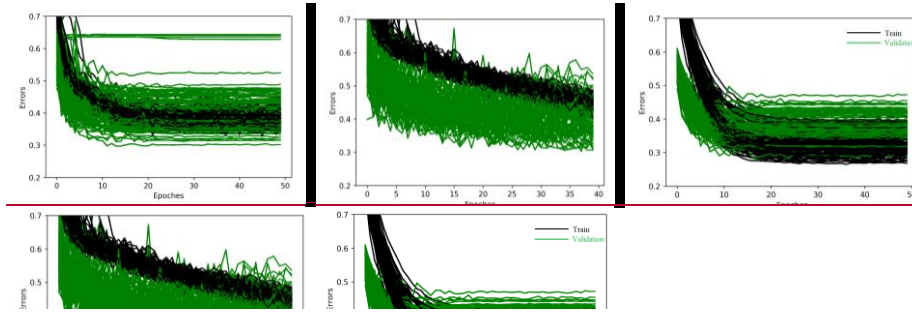
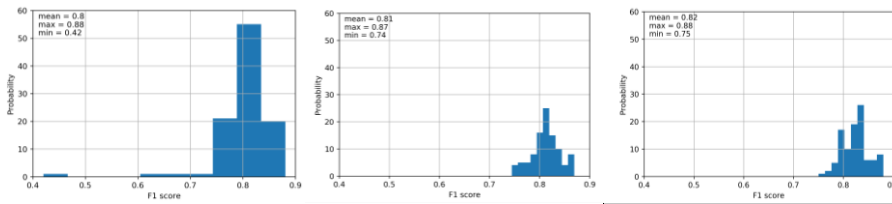



Figure 1714. Histograms of F1 scores of validation for CNN (left), LSTM (middle), and CNCNN-LSTM (right) models with 100 random train-validation splits.

640 ~~7~~

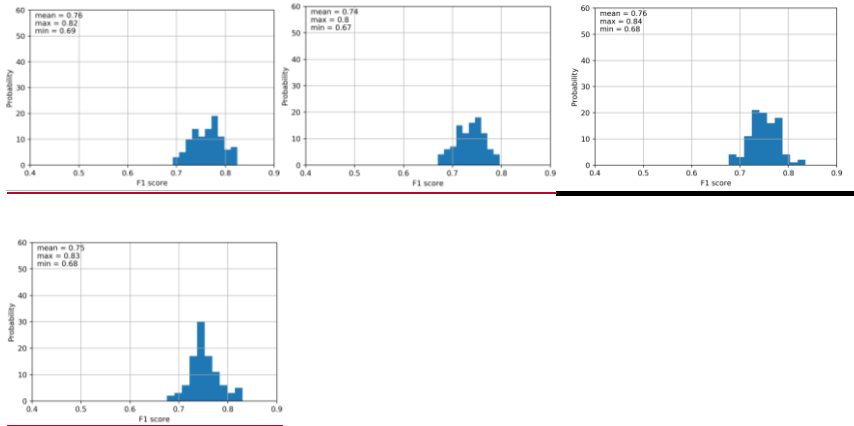



Figure 15. Histograms of F1 scores of validation for SVM (top left), DT (top middle), KNN (top right), and MLP
641 **(bottom left) models with 100 random train-validation splits.**

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

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

643
644 **Table 10. F1 scores of validation for SVM, DT, and KNN and MLP models with 100 random train-validation splits.**

Models	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

645 **3.~~3~~2.2 Number of parameters and run time**

646 The total number of parameters in CNN, LSTM, and ~~CN~~CNN-LSTM networks are 371586, 207874, and 664746,

647 respectively. The best performance has resulted from ~~CN~~CNN-LSTM with the highest number of parameters. Even

648 though the number of parameters for the LSTM model is less than CNN, the LSTM model shows better validation

649 performance. Furthermore, the number of parameters in the ~~CN~~CNN-LSTM model is much higher than the two other

models, but the computation time is not much higher. All three models take less than 24 hours to train with 100 shuffle splits for training and validation. The models are run on a CPU with four cores, 3.4 GHz clock speed, and 12 GB RAM.

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.

3.43 Order of input variables

AlthoughIt is not clear that whether the order of input variables in the input file is important throughmight influence multivariate TSC or not when using 2-D filters and 2-D max-pooling layers, there is no guideline for this order for multivariate TSC. In the benchmark, we randomly used this order from left to right: precipitation, minimum temperature, maximum temperature, net radiation, ATDD, AFDD, and snow depth. We randomly changed this order and applied the new order: snow depth, maximum temperature, precipitation, AFDD, net radiation, minimum temperature, and ATDD. Both models yielded the same average and minimum F1 scores, whereas the maximum F1 score from the order in the benchmark model (0.88) is higher than that of the second-order (0.86). Therefore, it can be concluded that the order does not significantly impact the results.

3.5 Generalization4 Testing

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

The results of the test models show that CNCNN-LSTM modelsmodel represent the best F1 score of 0.9192 (Table 811). Tables 79 and 811 show that although LSTM has slightly better validation performance, CNN works a little better in generalization by only 1 %. The better generalization of CNN can be becauseand LSTM is a little underfitted as LSTM models performed the same in testing.

The results of machine learning models are often harder to regularize, agreeingfor testing presented in Table 12 indicate that among the machine learning models KNN yields the best results with previous studies (e.g., Devineau et al., 2018, June).F1 scores of 78%. Tables 11 and 12 declare that deep learning models work much better than machine learning models for testing with 14% comparing CNN-LSTM with KNN as the best deep learning and machine learning models, respectively.

Table 811. Test F1 scores for LSTM, CNN, and CNCNN-LSTM models.

Models	F1 score
CNN	0.80

LSTM	0.~~79~~80
~~CN~~CNN-**LSTM**	0.~~91~~92

683

684

685 **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

686

687 **3.~~6~~5 Model comparison**

688 Multiple combined classifiers can be considered for pattern recognition problems to reduce errors as different

689 classifiers can cover internal weaknesses of each other (Parvin et al., 2011). The ~~ensemble~~combined classifier may be

690 less accurate than the most accurate classifier. However, the accuracy of the combined model is always higher than

691 the average accuracy of individual models. Combining two models improved our results compared to convolution-

692 only or LSTM-only networks in both training and ~~generalization.~~testing, supporting the previous studies (e.g., Sainath

693 et al., 2015). It can be because the ~~CN~~CNN-LSTM model incorporates both the temporal dependency of each variable

694 by using LSTM networks and the correlation between variables through CNN models. The combined CNN-LSTM

695 model efficiently benefit from automatic feature learning by CNN plus the native support for time series by LSTM.

696 ~~The~~ Although LSTM performed slightly better ~~generalization results from~~than CNN ~~compared to LSTM can be~~

697 ~~because of~~in validation, these models showed the ~~ability of~~same performance in testing. The CNN is able to partially

698 include both temporal dependency and the correlation between variables by using 1D and 2D filters, respectively~~,~~

699 ~~while~~. Although the LSTM is unable to incorporate the correlations between variables, it gives promising results with

700 relatively small dataset and captures longer temporal dynamics, while the CNN only captures temporal dynamics

701 within the length of its filters.

702 ~~4 Conclusion~~

703 ~~This project is a part of a project called DAVE, which aims to develop a tool to provide regional ice jam watches and~~

704 ~~warnings, based on the integration of three aspects: the current conditions of the ice cover; hydrometeorological~~

705 ~~patterns associated with breakup ice jams; and channel predisposition to ice-jam formation. The outputs of the previous~~

706 ~~tasks will be used to develop an ice-jam monitoring and warning module and transfer the knowledge gained to end-~~

707 ~~users to manage the risk of ice jams better.~~

708 ~~While most TSC research in deep learning is performed on 1D channels (Hatami et al., 2018, April), we propose deep~~

709 ~~learning frameworks for multivariate TSC for ice-jam prediction. The main finding from the comparison of results is~~

710 ~~that the CN-LSTM model is superior to the CNN-only and LSTM-only networks in both training and generalization~~

711 ~~accuracy, supporting the previous studies (e.g., Sainath et al., 2015). Though the LSTM network demonstrates quite~~

712 ~~good performance, the CNN model performed slightly better generalization, which agrees with previous studies (e.g.,~~
713 ~~Brunel et al., 2019).~~
714 ~~To our best knowledge, this study is the first study introducing these deep learning models to the problem of ice-jam~~
715 ~~prediction.~~ Even though our training data in supervised ice-jam prediction is small, the results reveal that deep learning
716 techniques can give accurate results, which agrees with a previous study conducted by Ordóñez and Roggen (2016)
717 in activity recognition. The excellent performance of CNN and ~~CN~~CNN-LSTM models may be partially due to the
718 characteristic of CNN that decreases the total number of parameters which does training with limited training data
719 easier (Gao et al., 2016, May~~) and including the correlation between involved variables.~~). However, our models will
720 be improved in the future by a larger dataset.

721 Among the developed machine learning models, SVM showed the best performance in validation, whereas KNN
722 worked the best in testing. However, the performance of deep learning models is much better than machine learning
723 models in both validation and testing. The machine learning models do not consider correlations between variables.
724 However, it is not the only reason that deep learning models worked better than machine learning models. As the
725 LSTM also does not consider correlations between variables but worked better than machine learning models. Some
726 characteristics of developed deep learning models can explain their better performance compared to machine learning
727 models. For instance, deep learning models perform well for the problems with complex-nonlinear dependencies, time
728 dependencies, and multivariate inputs.

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

732 **3.6 Discussion on the interpretability of deep learning models**

733 Even though the developed deep learning models performed pretty well in predicting ice jams in Quebec, the
734 interpretability of the results with respect to the physical processes of the ice jam is still essential. It is because although
735 deep learning models have achieved superior performance in various tasks, these really complicated models with a
736 large number of parameters might exhibit unexpected behaviours (Samek et al., 2017 & Zhang et al., 2021). This is
737 because the real-world environment is still much more complex. Furthermore, the models may learn some spurious
738 correlations in the data and make correct predictions with the 'wrong' reason (Samek and Müller, 2019). Hence,
739 interpretability is especially important in some real-world applications like flood and ice-jam predictions where an
740 error may cause catastrophic results. Also, interpretability can be used to extract novel domain knowledge and hidden
741 laws of nature in the research fields with limited domain knowledge (Alipanahi et al., 2015) like ice-jam prediction.
742 However, the nested non-linear structure and the "black box" nature of deep neural networks make interpretability of
743 their underlying mechanisms and their decisions a significant challenge (Montavon et al., 2018, Zhang et al., 2021
744 and Wojtas and Chen, 2020). That is why, interpretability of deep neural networks still remains a young and emerging
745 field of research. Nevertheless, there are various methods available to facilitate understanding of decisions made by a
746 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 also for eastern parts of Ontario and western New Brunswick. For other locations, the developed models can be transferred via re-training and a small amount of fine-tuning using labeled instances, rather than building from scratch. It is because the logic in the model may be transferable to the other sites with small modifications. To transfer a model from one river basin to another, historic 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 each site.

4 Conclusion

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

To our best knowledge, this study is the first study introducing these deep learning models to the problem of ice-jam prediction. The developed models are promising to be used to predict future ice jams in Quebec and in other 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 break upbreakup 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 availability as recorded ice jams are small which causes deep learning models to easily overfit to small number of 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 hydro-meteorological variables are not the only drivers of ice-jam formation. The geomorphological indicators that control the formation of ice jams include the river slope, sinuosity, a barrier such as an island or a bridge, narrowing of the channel, and confluence of rivers. In the future, a geospatial model using deep learning will be developed to examine the impacts of these geospatial parameters on the 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

782 of the paper with conceptual edits from Karem Chokmani and Saeid Homayouni. Yves Gauthier and Simon
783 Tolszczuk-Leclerc helped in the refinement of the objectives and the revision of the methodological developments.

784 **Acknowledgment**

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

788 **References**

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

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

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

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

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

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

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

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

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

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

808 Brownlee, J. (2017). A gentle introduction to exploding gradients in neural networks. Retrieved from
809 https://machinelearningmastery.com/exploding-gradients-in-neural-networks/.2018). Deep learning for time series
810 forecasting: predict the future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery.

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

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

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

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

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

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

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

825 del Campo, F. A., Neri, M. C. G., Villegas, O. O. V., Sánchez, V. G. C., Domínguez, H. D. J. O., & Jiménez, V. G.
826 (2021). Auto-adaptive multilayer perceptron for univariate time series classification. *Expert Systems with*
827 *Applications*, *181*, 115147.

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

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

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

836 Données Québec: Historique (publique) d'embâcles répertoriés au MSP - Données Québec,. Retrieved from
837 https://www.donneesquebec.ca/recherche/dataset/historique-publique-d-embacles-repertories-au-msp. (last access:
838 15 June 2021).

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

Gamboa,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.

J. C. B. (2017). Deep learning for time-series analysis. arXiv preprint arXiv:1701.01887.

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 on artificial intelligence and statistics (pp. 249-256). JMLR Workshop and Conference Proceedings.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). Cambridge: MIT press.

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 on computer vision (pp. 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 on machine learning (pp. 448-456). PMLR.

Jordan, J. (2018). Setting the learning rate of your neural network. Retrieved from https://www. jeremyjordan. me/nn-learning-rate.

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 on machine learning (pp. 2342-2350). PMLR.

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). Multivariate lstm-fcns for time series classification. Neural Networks, 116, 237-245.

Karpathy, AKashiparekh, 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).

. (2017). Convolutional neural networks for visual recognition. Retrieved from http://cs231n.github.io/convolutional-networks/.

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.

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 on artificial intelligence and computer applications (ICAICA) (pp. 352-355). IEEE.

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

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

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

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

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

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

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

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

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

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

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

926 Nosratabadi, S., Mosavi, A., Duan, P., Ghamisi, P., Filip, F., Band, S. S., ... & Gandomi, A. H. (2020). Data science
927 in economics: comprehensive review of advanced machine learning and deep learning methods. Mathematics, 8(10),
928 1799.

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

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

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

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

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

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

942 Prowse, T. D., & Bonsal, B. R. (2004). Historical trends in river-ice break-up: a review. Hydrology Research, 35 (4-
943 5), 281-293.

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

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

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

949 Rodríguez, J. J., & Alonso, C. J. (2004, December). Support vector machines of interval-based features for time series
950 classification. In International Conference on Innovative Techniques and Applications of Artificial Intelligence (pp.
951 244-257). Springer, London.

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

955 Samek, W., & Müller, K. R. (2019). Towards explainable artificial intelligence. In Explainable AI: interpreting,
956 explaining and visualizing deep learning (pp. 5-22). Springer, Cham.

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

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

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

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

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

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

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

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

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

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

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

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

983   White, K. D. (2003). Review of prediction methods for breakup ice jams. Canadian Journal of Civil Engineering,
984   30(1), 89-100.

985   Wong, S. C., Gatt, A., Stamatescu,White, K. D., & Daly, S. F. (2002, January). Predicting ice jams with discriminant
986   function analysis. In ASME 2002 21st International Conference on Offshore Mechanics and Arctic Engineering (pp.
987   683-690). American Society of Mechanical Engineers.

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

989   Wu, J., Yao, L., & Liu, B. (2018a, April). An overview on feature-based classification algorithms for multivariate
990   time series. In 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA) (pp.
991   32-38). IEEE.

992   , V., & McDonnell, M. D. (2016, November). Understanding data augmentation for classification: when to warp?.
993   In 2016 international conference on digital image computing: techniques and applications (DICTA) (pp. 1-6). 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 on Multimedia (pp. 461-470).

Zaremba, W., Sutskever, 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., & Vinyals, O. (2014). RecurrentChen, 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 on artificial intelligence.

Yi, S., Ju, J., Yoon, M. K., & Choi, J. (2017). network regularization.Grouped convolutional neural networks for multivariate time series. arXiv preprint arXiv:1409.23291703.09938.

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.

1025