

Predicting Stock Prices Using Deep Learning Models: CNN, LSTM, and CNN-LSTM. A Case Study of NVIDIA Stock Price Prediction for the Period 2020-2024

Rabia Bousbia Laiche

rabia-bousbia.laiche@univ-eloued.dz

Mohammed Lassoued

lassoued.mohammed@gmail.com

Received: 20/07/2024

Accepted: 12/09/2024

Published: 18/09/2024

Abstract

This paper addresses the significance of stock price prediction for investors and the growing application of deep learning techniques, particularly advanced Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, in forecasting stock prices. We propose a hybrid model, referred to as the Long Short-Term Memory-Convolutional Neural Network (LSTM-CNN) model, which integrates features extracted from different representations of the same dataset, specifically stock time series, to predict stock prices. The proposed model employs a CNN to extract features from the data and an LSTM network to retain temporal information and capture complex relationships between time points. The performance of the proposed model is evaluated against individual models and the hybrid model (CNN, LSTM, and LSTM-CNN) using data from Nvidia during the period from 01/01/2020 to 01/08/2024. Our LSTM-CNN feature integration model demonstrates superior performance in predicting stock prices compared to the individual models. Consequently, this study reveals that prediction errors can be efficiently minimized by combining both temporal and local features from the same dataset, rather than utilizing these features independently.

Keywords: Stock price prediction, Nvidia, deep learning, neural networks, CNN models, LSTM, LSTM-CNN.

1. Introduction

The method of analyzing the historical price series of a stock is based primarily on studying its movement through charts for the purpose of predicting price trends. The key hypothesis on which this prediction can be based is that "history repeats itself," meaning the current price of an asset always reflects all previous information immediately, i.e., the stock's behavior is embedded in its historical data.

On the other hand, there is also the random walk hypothesis, which claims that stock prices change independently of their history. In other words, tomorrow's price will depend only on tomorrow's information, regardless of today's price. These two hypotheses indicate that there is no precise way to predict stock prices (2).

The field of methods for predicting stock behavior and modeling it involves the interaction of a variety of disciplines, including economics, statistics, mathematics, and computer science. Notably, in 2012, it was estimated that nearly 85% of trades in the U.S. stock markets were executed by algorithms (3).

When using econometrics—applying statistical methods to economic data to test theories, extract relationships, and predict future trends—it is useful to have interpretive power, as this power draws conclusions based on theoretical background. However, the assumptions used in standard econometric models do not necessarily hold in the real world due to the limitations of the assumptions, inability to predict exceptional events, influence of qualitative factors, and increasing complexity of global markets, among others. In contrast, Artificial Neural Networks (ANNs) such as those used in financial analysis, including traditional neural networks, Recurrent Neural Networks (RNNs) like the LSTM model, and Convolutional Neural Networks (CNNs), are not bound by these assumptions and can detect non-linear relationships in data properties. Therefore, many researchers have been studying stock price prediction based on these models.

Kimoto et al. (4) proposed a prediction system for forecasting stocks listed on the Tokyo Stock Exchange using an artificial neural network system and achieved excellent profits using the prediction system in a simulation exercise. Kim et al. used a genetic algorithm approach to optimize the threshold for distinguishing between features and the connection weights between layers to predict stock price indices. The results indicated that this approach outperformed traditional neural networks (5). Tsang et al. proposed a buy-sell alert system using artificial neural networks to predict stock prices in Hong Kong (6). This system showed an overall success rate of more than 70%.

In his study on air quality prediction, which can provide a theoretical basis for environmental protection management and decision-making, Zhang J. (7) proposed a CNN-LSTM model to improve air quality prediction accuracy. First, the CNN feature extraction function was used to extract features from the data. Then, feature vectors in the form of a sequence were created, which were passed to an LSTM network. The LSTM layer learned the changing rules of air quality data to predict future data. Taking the air quality index in Beijing as an example, the prediction results of the CNN-LSTM model were compared with those of the Autoregressive Moving Average (ARMA), the Seasonal Autoregressive Integrated Moving Average (SARIMA), the Recurrent Neural Network (RNN), and the Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) models. The results show that compared to other individual prediction models, the CNN-LSTM model achieved the highest prediction accuracy. In particular, the CNN-LSTM model was compared with the SARIMA model, a representative time series model. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) of CNN-LSTM were reduced by 3.17% and 5.46%, respectively, and the R-squared value improved by 8.45%.

In Li J.'s (8) study on influenza, which has become a significant global health challenge, accurate prediction of influenza activity is critical to mitigating its impact. Li aimed to develop a hybrid CNN-LSTM model to predict the Influenza-Like Illness (ILI) rate in Hebei Province, China, providing more accurate guidance for influenza prevention and control measures.

ILI% data from 28 national hospitals in Hebei Province, spanning from 2010 to 2022, were used to develop the CNN-LSTM model through the PyTorch deep learning framework. Additionally, R and Python were used to develop four other models commonly used for predicting infectious diseases. Retrospective predictions were then made, and the predictive performance of each model was compared using metrics such as MAE, RMSE, and MAPE.

Based on historical data of influenza and infection rates (ILI%) from 28 national hospitals in Hebei Province, the SARIMA, XGBoost, CNN, LSTM, and CNN-LSTM models were created. In the test set, all models effectively predicted influenza and infection rate trends (ILI%). Among these, the CNN-LSTM model showed the best predictive performance, followed by XGBoost, LSTM, CNN, and finally the SARIMA model, which demonstrated the least favorable performance.

In recent years, deep learning techniques have achieved remarkable results in time series forecasting, particularly CNNs and LSTMs, which have demonstrated excellent performance in processing temporal data. Deep learning models can automatically learn from data characteristics and patterns and are well-suited for handling nonlinear and high-dimensional time series data (9). As a result, they are widely used in financial time series analysis, such as stock price prediction, crude oil price forecasting, carbon trading price prediction, and other fields. These models have their own memory and can offer relatively accurate predictions (10). Typically, these models are combined with optimization algorithms and decomposition integration methods to improve the accuracy and convergence rate of predictions (11).

Although deep learning is expected to improve prediction accuracy, it may be limited when dealing with more complex problems. Nonetheless, experimental results from various studies have shown that the CNN-LSTM model outperforms standard deep learning models in terms of prediction accuracy and stability. Both CNN and LSTM exhibit strong feature extraction capabilities from data. CNN can learn the internal representation of time series data directly without needing manual feature extraction, while LSTM has a strong ability to extract temporal features and capture implicit information in time series trends more accurately. According to a sample of these studies, the combination of CNN and LSTM models outperforms the ARIMA model in stock price prediction (12). The CNN-LSTM model was used to predict the air quality index in Beijing and was compared with other traditional baseline models, showing better predictive ability and improved robustness. Similarly, the CNN-LSTM model predicted household energy consumption (13), achieving near-perfect performance in forecasting previously difficult-to-predict energy consumption. Wang et al. proposed an enhanced CNN-LSTM model with an attention mechanism that focuses on the most important parts of the data, improving prediction accuracy by assigning greater weight to significant information (14). This method was applied to short-term load

forecasting for cogeneration, addressing the problem of information loss resulting from long time-series data input. Consequently, it improved the accuracy, robustness, and generalization ability of the energy load forecast. These studies have demonstrated the effectiveness of the CNN-LSTM model across various fields (15).

The aim of this article is to identify models with high predictive accuracy and robustness by comparing different sets of prediction models. The performance of these models is evaluated using MSE, MAE, and R^2 metrics to investigate the differences in prediction accuracy. This research contributes by comparing the individual CNN model and the individual LSTM model with the combined CNN-LSTM model to examine the accuracy of predicting the company's future stock price and its ability to capture price spikes over time.

2. Experimental Study:

2.1 Data and Work Environment

In this study, we use daily stock data from Nvidia, which has experienced significant gains in the stock market. We collect data on high, low, open, close, and trading volume prices, covering 1,152 data points from January 1, 2020, to August 1, 2024, from the Yahoo Finance database. This period witnessed a substantial increase in stock prices due to rising demand for AI-supporting semiconductor chips from Nvidia.

The last recorded price of a stock during the day is called the closing price. This is the standard price considered when analyzing financial time series. However, this price must be adjusted according to corporate actions before any analysis of historical data is conducted. These actions could include stock splits, dividends, or rights offerings. All these actions affect the "nominal" price of the stock, although none impact its "real" price. For proper model training, the effects of these actions must be removed from the nominal stock price. The closing price after these adjustments is called the adjusted close price (Adj-Close), which we have used in our analysis.

We assigned a training and testing dataset. We use 922 data points as training data, representing 80% of the total, and 230 data points as test data.

Data Loading: Data (time series) can be loaded from the desktop in xls or csv format or can be directly retrieved from Yahoo Finance after installing the yfinance library. We followed this method to load Nvidia's time series data for the period from 01/01/2020 to 01/08/2024. The first elements of the data are shown as follows:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2020-01-02	158.779999	160.729996	158.330002	160.619995	153.938202	22622100
1	2020-01-03	158.320007	159.949997	158.059998	158.619995	152.021393	21116200
2	2020-01-06	157.080002	159.100006	156.509995	159.029999	152.414322	20813700
3	2020-01-07	159.320007	159.669998	157.320007	157.580002	151.024658	21634100
4	2020-01-08	158.929993	160.800003	157.949997	160.089996	153.430252	27746500

Work Environment: Google Colab and Importing Necessary Libraries:

Google Colab is an enhanced cloud-based version of Jupyter Notebook, designed for writing and executing code, as well as managing Notebook documents (a document where you can write code such as Python scripts, add text, and images, and execute them to display the results). It provides free access to GPUs and TPUs, which are used for building machine learning or deep learning models. One of its key features is that the libraries are pre-installed, in addition to supporting GPU acceleration by leveraging Google's powerful infrastructure. Lastly, a significant feature is the integration of **Gemini**, Google's most powerful AI model to date, into Google Colab. This offers a major advantage to programmers by enabling error correction through this addition. Recently, it has also become possible to generate commands through direct textual instructions (generate) for beginners or input instructions using Python code (coding) directly for experts.

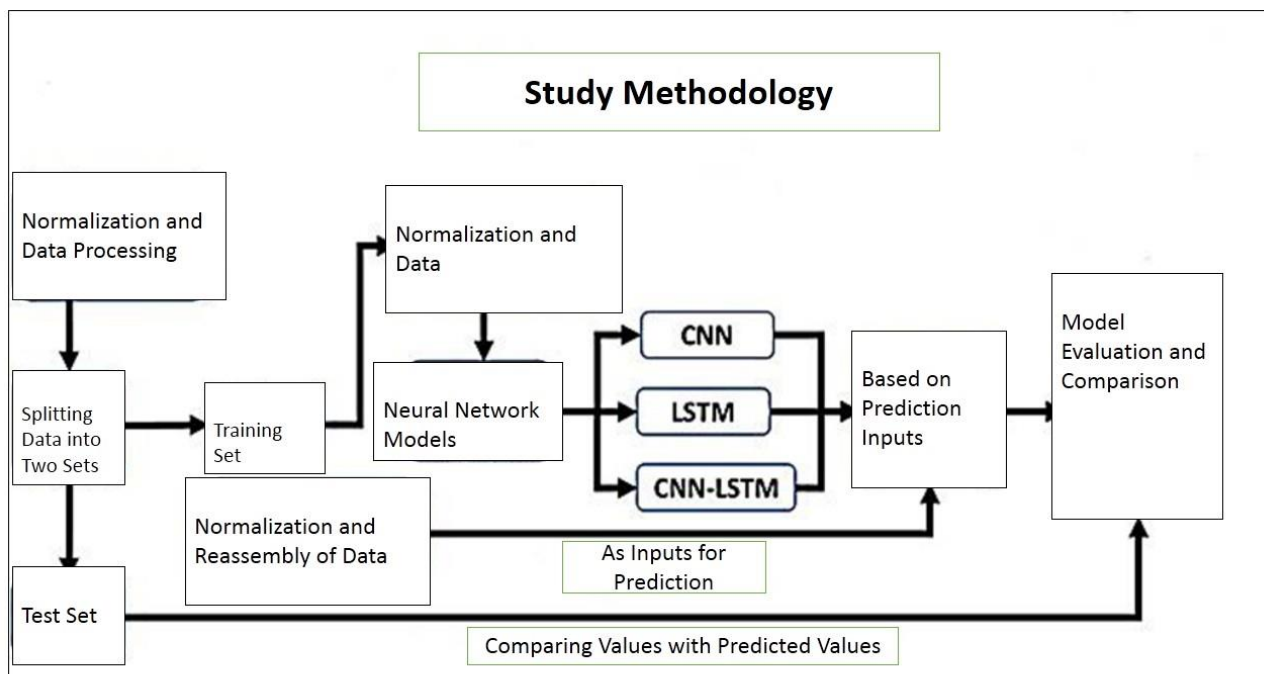


Start coding or generate with AI.

We call the following libraries: **numpy**, **sklearn.preprocessing**, **keras.models**, **keras.layers**, which are necessary for the following tasks:

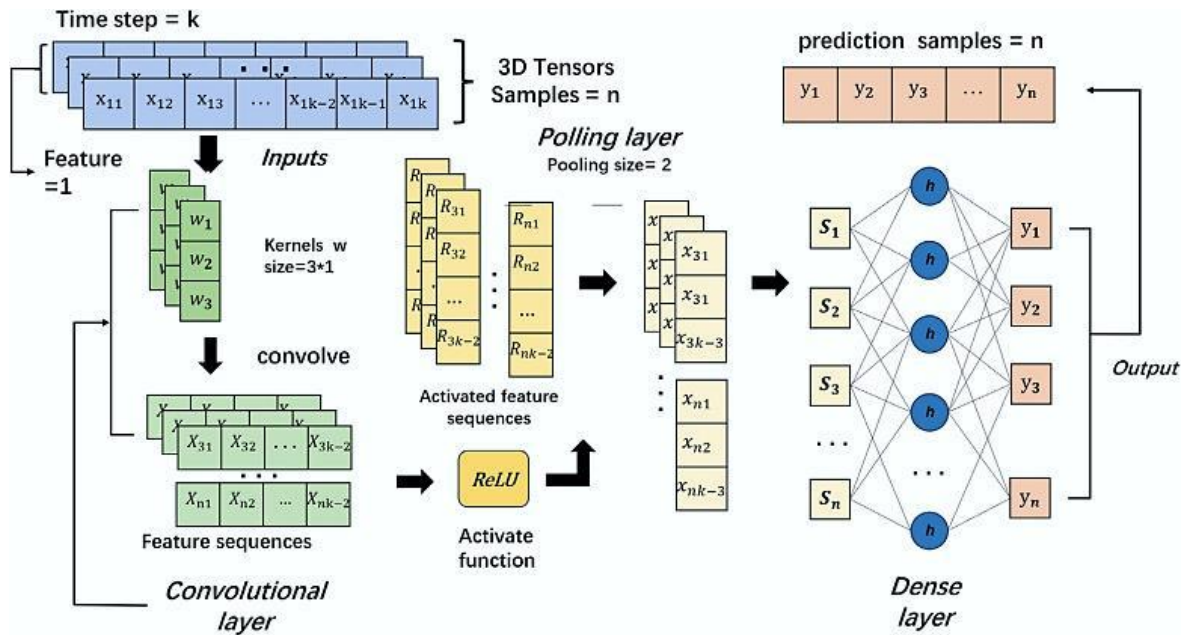
- **Numpy:** Provides support for mathematical and scientific operations on arrays. It is widely used for fast and efficient computations on numerical data, especially for matrix manipulation and mathematical operations like linear algebra and statistics.
- **sklearn.preprocessing:** Offers tools for data preprocessing before using them in models. It is used for data normalization, such as StandardScaler and MinMaxScaler, which normalize features by rescaling values to a certain range, typically [0, 1] or any other range specified. It also transforms data, such as converting categorical data to numerical using LabelEncoder.
- **keras.models:** Facilitates the easy and efficient creation of deep learning models, especially with Sequential models. It also enables model compilation, training, and evaluation.
- **keras.layers:** Provides various types of layers to build neural networks, allowing the addition of layers like Conv1D, MaxPooling1D, Flatten, and Dense to models. Layers can be customized with various parameters like the number of units, activation type, etc.

2-2 Study Approach and Model Structure



2-2-1 CNN Model: Concept and Design

A **Convolutional Neural Network (CNN)** is a traditional deep learning model primarily used for image recognition and computer vision applications but can also be applied to time series data analysis (16). This model belongs to the family of **Multi-Layer Perceptrons (MLP)**, a type of neural network widely used in machine learning and artificial intelligence. An MLP is a feed-forward neural network, meaning that information flows in one direction from the input layer to the output layer (17).



The MLP structure consists of three or more layers: an **Input layer**, one or more **Hidden layers**, and an **Output layer**. The input layer is fed with our input data, and we obtain the results from the output layer. We can increase the number of hidden layers to make the model more accurate and complex depending on the task at hand (18).

CNN Architecture: CNN models are highly effective at extracting local features and can be used for time series forecasting and analysis. A CNN model consists of **convolutional layers**, **pooling layers**, **activation functions**, and **dense layers** (as shown in the diagram below).

This diagram illustrates the architecture of a CNN designed to process sequential data, which consists of the following main components:

1. **Inputs:**
 - Comprised of **Tensors** with k time steps and n samples. Typically, it takes the form of a 2D matrix (time \times features).
 - Each sample has one feature (Feature = 1).
2. **Convolutional Layer:**
 - Uses **filters** (Kernel) of size 3×1 on the data to extract features, aiming to identify local patterns in the data. Several consecutive convolutional layers can be used to achieve this goal.
 - Produces **Feature sequences**.
 - **Activation function:** Uses the **ReLU** function to activate the extracted features.
3. **Pooling Layer:**
 - At this stage, the data dimensions are reduced to help make the model less sensitive to small changes. Common types of pooling include **Max Pooling** and **Average Pooling**. In this diagram, pooling size = 2 is used.
4. **Dense Layer (Fully connected):**
 - A fully connected layer that links the pooled features to the outputs.
 - Contains h hidden units.

5. Outputs:

- Produces n prediction samples from y_1 to y_n .

This architecture enables the network to process sequential data, extract relevant features, and make predictions. The convolutional layer identifies local patterns, the pooling layer reduces data dimensions, and the dense layer aggregates information to produce the final predictions.

2-2-2 LSTM Model:

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) widely used for time series prediction. They were first introduced by researchers Sepp Hochreiter and Jürgen Schmidhuber in 1997.

The **LSTM model** is widely used in deep learning tasks such as natural language processing, speech recognition, and time series analysis. It is specifically designed to address the **vanishing gradient problem** that occurs in traditional RNNs, which makes it difficult for the network to learn long-term dependencies in sequential data.

To understand how the **LSTM model** works, it is important to first grasp the concept of a traditional **RNN**. In a typical RNN, the output of the previous time step is fed back into the network as input for the current step. This allows the network to retain some form of memory and capture sequential information. However, traditional RNNs suffer from the vanishing gradient problem, where the gradients used to update the network weights diminish significantly as they propagate through time, making it hard for the network to learn long-term dependencies.

LSTM networks are capable of handling long-term dependencies in data, which traditional RNNs struggle with.

The LSTM model consists of the following cells:

- **Memory cell:** Stores information from previous steps.
- **Forget gate:** Controls the information that is forgotten from the memory cell.
- **Input gate:** Controls the information added to the memory cell.
- **Output gate:** Controls the information outputted from the memory cell.

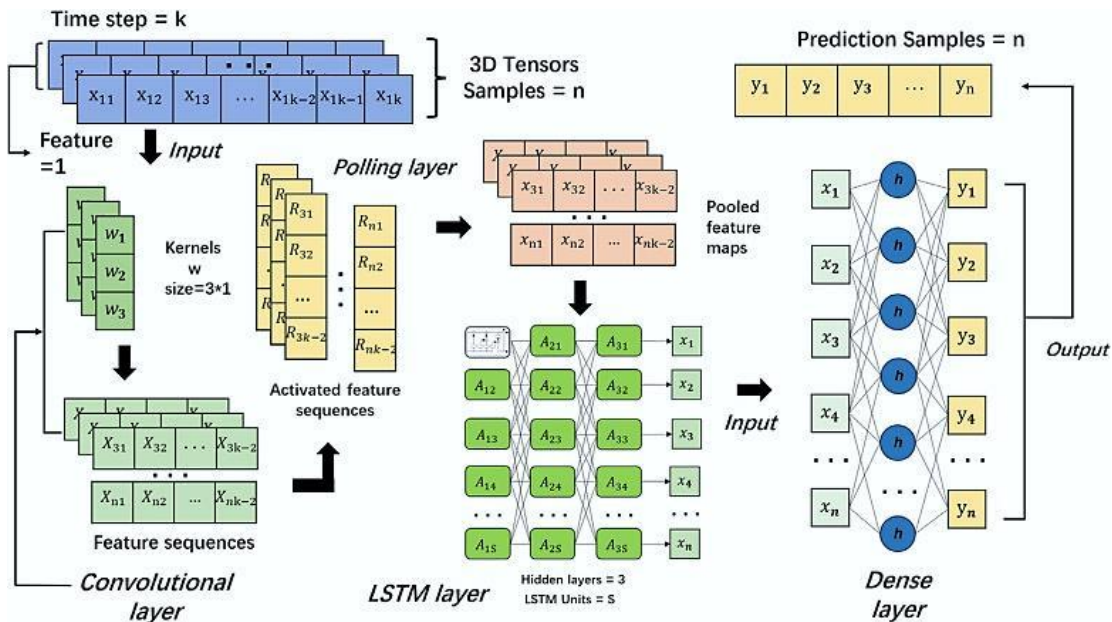
Advantages of the LSTM model:

- **Handles long-term dependencies:** LSTM networks can handle long-term dependencies in data, which traditional RNNs cannot.
- **High accuracy:** LSTM networks have shown high accuracy across various applications.
- **Versatility in applications:** LSTM networks can be applied to a wide range of tasks, including time series prediction.

2-2-3 CNN-LSTM Model: Concept and Design

The **hybrid CNN-LSTM model** combines the **local feature extraction** capability of CNNs with the **long-term dependency handling** ability of LSTMs. The figure below shows the architecture of our CNN-LSTM model, which consists of convolutional layers, pooling layers, LSTM layers, a **sigmoid activation** function, and a dense layer.

The convolutional layers use 1D kernels to extract features from the original data, converting them into **feature maps**. The pooling layer selects and reduces the dimensions of the feature maps, which are then decoded by the LSTM layers and fed into the dense layer to obtain the model's output. This process generates the following formula:



This diagram illustrates the structure of a complex neural network that combines a **Convolutional Neural Network (CNN)** with a **Long Short-Term Memory (LSTM)** network, consisting of the following key elements:

1. **Inputs:**

- 3D data tensors with k time steps and n samples.
- Each sample has a single feature (Feature = 1).

2. **Convolutional Layer:**

- Uses kernels of size 3×1 to perform convolution on the inputs.
- Produces **feature sequences**.
- **Pooling Layer:** Aggregates the activated features.

3. **LSTM Layer:**

- Consists of 3 hidden layers.
- Each layer contains 5 LSTM units.
- Processes the aggregated features to extract temporal relationships.

4. **Dense Layer:**

- A fully connected layer that links the LSTM outputs to the final predictions.
- Contains h hidden units.
- **Outputs:** Produces n prediction samples, from y_1 to y_n .

This complex architecture combines the strength of **CNNs** in extracting spatial features with the ability of **LSTMs** to handle temporal sequences. The convolutional layer extracts local patterns, while the pooling layer reduces the dimensionality of the data. The LSTM layers then process these features to capture long-term temporal relationships. Finally, the dense layer aggregates all this information to produce the final predictions.

This type of network is particularly effective in handling complex sequential data, such as time series prediction. The advantage of this model is that it can support very long input sequences, which are read as blocks or subsequences by the CNN and then aggregated by the LSTM.

Combining these two types of networks can enhance performance in time series classification tasks. The main methods of combining them are:

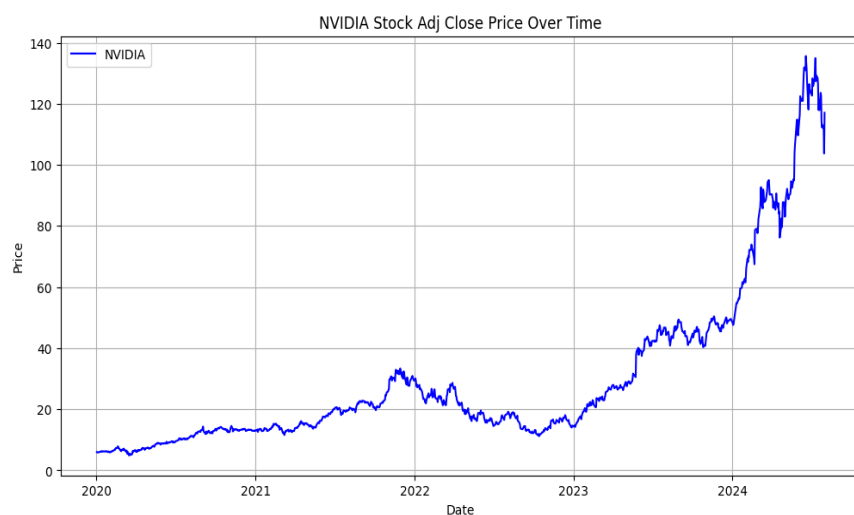
- **Using CNN outputs as inputs to LSTM:** This allows the LSTM to learn features from input data that the CNN has extracted.
- **Using LSTM outputs as inputs to CNN:** This allows the CNN to learn features from LSTM outputs.
- **Using a parallel architecture:** Both CNN and LSTM process the input data independently, and their outputs are combined and passed to the fully connected layer.

The choice of method depends on the specific problem being solved and the characteristics of the input data. It may be necessary to experiment with different architectures and hyperparameters to find the best model for the problem at hand (19).

In our research, we propose using a hybrid neural network with a **CNN** and **LSTM** to predict stock prices using adjusted closing prices as model inputs. The **CNN** is used to extract features, while the feature vectors are fed into the **LSTM** for training and short-term stock price prediction. Based on previous studies, we believe this model can improve both the accuracy of value prediction and the trend compared to a single-architecture neural network.

3. Results and Discussion

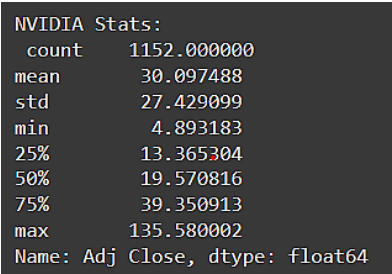
3-1 Graphical Analysis:



This chart represents the evolution of the adjusted close price (Adj Close) for Nvidia stock over a specific time period. It shows how the stock price has changed over time, providing an overview of its performance. The graph can be divided into three stages:

1. **Stage One (2020-2021):** This phase indicates relative stability in the stock price, with no significant fluctuations.
2. **Stage Two (2021-2022):** During this period, the stock experienced noticeable fluctuations and a significant increase in price.
3. **Stage Three (2023 to mid-2024):** This phase saw a rapid rise in price due to increased demand for Nvidia's tech products in fields like AI, gaming, and cryptocurrency mining. However, this sharp increase was followed by a slight decline, possibly indicating a market correction after a period of inflated prices.

3-2 Descriptive Analysis:



```
NVIDIA Stats:
count      1152.000000
mean       30.097488
std        27.429099
min         4.893183
25%        13.365304
50%        19.570816
75%        39.350913
max        135.580002
Name: Adj Close, dtype: float64
```

The statistics in the table above provide a summary of Nvidia's stock price data over the period covered. The results are as follows:

- **Count:** This represents the number of days on which stock prices were recorded, indicating data for 1,152 days.
- **Mean:** The average stock price over the time period is 30.097. This gives an idea of the overall price level during this period.
- **Standard Deviation:** The standard deviation (27.43) shows how much the prices deviate from the mean. A relatively high standard deviation indicates significant fluctuations in the stock price.
- **Minimum:** The lowest recorded stock price during the period is 4.893183.
- **25th Percentile (Q1):** 25% of stock prices were below 13.365304.
- **Median (50%):** The median stock price is 19.570816, which represents the middle value when data is ordered, giving a better sense of central tendency compared to the mean.
- **75th Percentile (Q3):** 75% of stock prices were below 39.350913, reflecting the upper half of the data distribution.
- **Maximum:** The highest recorded stock price during the period is 135.580002 USD.

Observations:

- **Price Range:** Stock prices ranged from 4.89 to 135.58 USD, reflecting significant variation over the studied period.
- **Market Fluctuations:** The high standard deviation suggests that the stock experienced substantial volatility, which is common for tech companies undergoing rapid growth followed by corrections.

Conclusion:

The data shows that Nvidia has undergone a volatile period, with significant fluctuations in stock prices. However, the overall trend is upward, as indicated by the comparison between the first and third quartiles and the maximum price. These statistics can help investors understand the extent of stock price volatility and make informed investment decisions.

3-2 Stock Price Prediction Using CNN Model

3-2-1 Building the CNN Model:

- **Data Preparation:**
 - **Data Collection:** Obtain the required dataset.
 - **Data Preprocessing:** Clean and format the data for model input.
 - **Data Transformation:** Convert the time series into slices of fixed length (windows) that can be fed into CNN layers.

- **Model Architecture:**
 - **Convolutional Layers:** Use Conv1D layers (instead of Conv2D) as time series data is usually one-dimensional. Filters are applied to detect patterns.
 - **Activation Layers:** Use activation functions like ReLU to add non-linearity to the model.
 - **Pooling Layers:** Use MaxPooling1D to reduce the dimensionality of the extracted features, improving model efficiency by reducing the number of parameters.
 - **Dropout Layers:** These are used to prevent overfitting by randomly turning off some neurons during training.
 - **Flatten and Dense Layers:** After the convolutional layers, a Flatten layer is added to convert the output into a flat structure, followed by a Dense layer for the final output.
 - **Compile:** The model is compiled using a loss function (MSE - Mean Squared Error) and RMSE (Root Mean Squared Error) as the evaluation metric, with **Adam** as the optimizer.
 - **Training:** Use the fit function to train the model on the training dataset, updating the weights through the optimization process. Data can be split into training and test sets.
 - **Evaluation:** The model's performance is tested using unseen data to evaluate its effectiveness.

3-2-2 Model Output:

- **Training the CNN:** After building the network, it is trained (with 100 epochs).

Epoch 96/100
29/29 ————— 0s 3ms/step - loss: 5.4701e-05
Epoch 97/100
29/29 ————— 0s 3ms/step - loss: 4.9697e-05
Epoch 98/100
29/29 ————— 0s 2ms/step - loss: 4.4983e-05
Epoch 99/100
29/29 ————— 0s 2ms/step - loss: 3.8026e-05

Model Training Results

These outputs show the results of training a neural network machine learning model, including the following information:

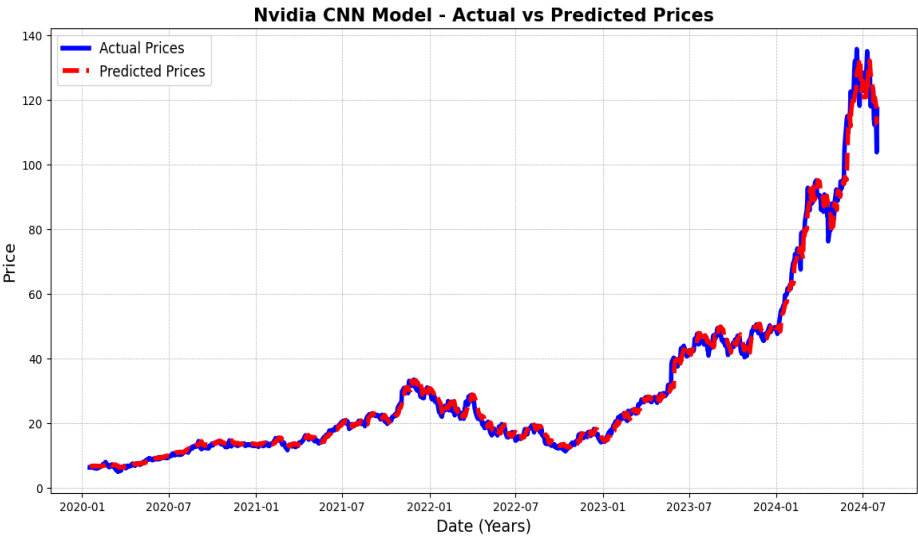
- **Epoch X/100:** Indicates the current iteration out of a total of 100 training epochs. Here, we see the final iterations from 96 to 99.
- **29/29:** Represents the number of batches completed out of the total batches per epoch. All batches have been completed.
- **0s 2ms/step:** Refers to the time taken to complete each epoch; each epoch took less than a second, with an average of 2 milliseconds per step.
- **loss: X.XXXXe-05:** Represents the loss value at the end of each epoch. We observe that the loss (Mean Squared Error) decreases with each epoch, indicating improved model performance.

Additional Notes:

- **Decreasing Loss Value:** The reduction in loss value indicates that the model is learning and improving with each epoch.

- **Very Small Loss Values:** Loss values in the range of $X.XXXE-05$ indicate excellent performance of the model on the training data.
- **Training Speed (2ms/step 0s):** Indicates that the model is relatively small or that the hardware used for training is powerful.

Graphical Representation of the Model for Original and Predicted Data



The chart compares the actual prices to the predicted prices of Nvidia stock using the CNN model. It shows:

- **Actual Prices (Blue Line):** The actual closing prices of Nvidia stock.
- **Predicted Prices (Red Line):** The closing prices predicted by the CNN model.

From the chart, it is evident that the model is capable of tracking the general trend of stock prices and capturing most of the significant fluctuations. The model demonstrates high accuracy, responsiveness to changes, and some minor deviations, especially during periods of high volatility. Overall, the CNN model proves to be a powerful tool for stock price prediction.

Model Evaluation

Evaluation is performed using the following metrics:

- Train RMSE: 0.0065
- Test RMSE: 0.0560
- Train MAE: 0.0046
- Test MAE: 0.0412
- Train R2: 0.9917
- Test R2: 0.9369

Interpretation of These Results:

- **RMSE = 0.0065:** This means that the average prediction error is \$0.281, with larger errors being weighted more heavily.
- **MAE = 0.0046:** Indicates that the average absolute difference between the predicted and actual values is \$0.230.

- **$R^2 = 0.9917$:** Shows that the model explains 99.17% of the variance in stock prices, which indicates excellent performance.
- **RMSE (Root Mean Squared Error):** Measures the square root of the average squared differences between actual and predicted values, reflecting the model's prediction accuracy.
- **MAE (Mean Absolute Error):** Measures the average magnitude of errors in predictions, without considering their direction.
- **R^2 (R-squared):** Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables, reflecting the goodness of fit of the model.

2-3 Predicting Stock Prices Using LSTM Model

To forecast stock prices, the following steps apply to the Nvidia stock prediction project or any other stock:

- **Data Preparation:**
 - Load historical stock price data for Nvidia.
 - Clean the data and handle missing values.
 - Normalize the data using MinMaxScaler or StandardScaler to transform it to the range [0,1].
 - Split the data into training and testing sets, with 80% for training and 20% for testing.
 - Convert the data into time sequences of a specified length (e.g., using 60 days of data to predict the next day) using the create_dataset function.
- **Building the LSTM Model:**
 - Create a Sequential model using Keras.
 - Add an LSTM layer with a suitable number of units.
 - Add a Dropout layer to prevent overfitting.
 - Add a Dense layer with a linear activation function to predict the closing price.
- **Training the Model:**
 - Train the model on the training data using the fit function.
 - Compile the model using the Adam optimizer and Mean Squared Error loss function.
 - Set the number of epochs to 100 and batch size to 32.
 - Train the model on the training data.
- **Evaluating the Model:**
 - Use test data to evaluate the model's performance.
 - Reverse normalize the data using scaler.inverse_transform.
 - Calculate performance metrics such as RMSE, MAE, and R^2 , and plot actual vs. predicted values.
- **Forecasting Future Prices:**
 - Load the data for the desired forecast period.

- Normalize the data using the same scaler used previously.
- Create a time sequence for forecasting and predict closing prices using the trained model.
- Reverse normalize the data.

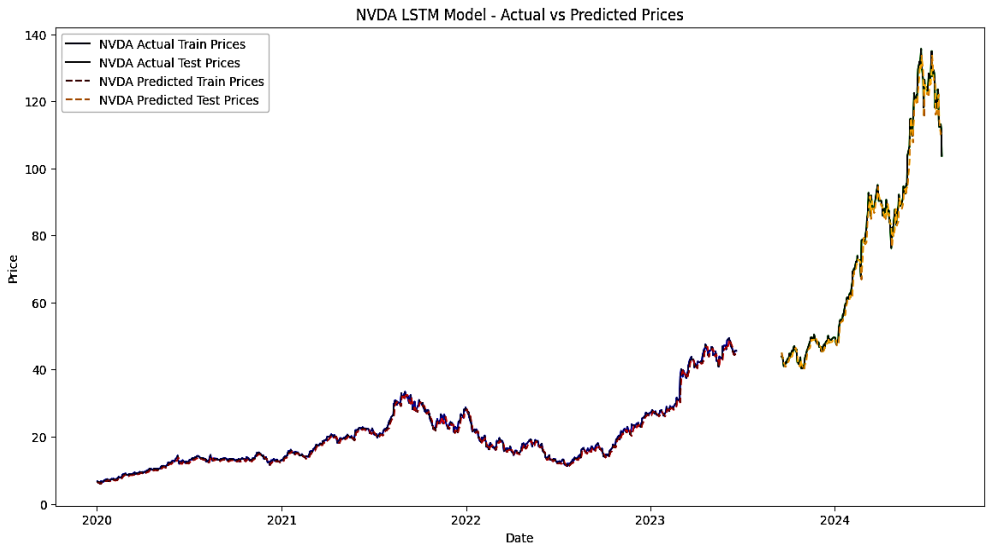
2-4 Model Outputs

- **Training the LSTM Neural Network:** After building the network, train it by repeating the process 100 times (Epoch=100).
- Epoch 97/100
- **29/29** ————— **1s** 10ms/step - loss: 4.8505e-05
- Epoch 98/100
- **29/29** ————— **0s** 10ms/step - loss: 4.8607e-05
- Epoch 99/100
- **29/29** ————— **1s** 11ms/step - loss: 5.3007e-05
- Epoch 100/100
- **29/29** ————— **1s** 11ms/step - loss: 4.9276e-05

Results of the LSTM Model Training

These outputs show the results of training a machine learning model for a neural network, characterized by a decrease in the Mean Squared Error (MSE) with each epoch, indicating an improvement in the model's performance.

- **Graphical Representation of the Model for Original and Predicted Data After Training:**



The figure displays a graph showing the results of the LSTM model for predicting NVDA stock prices. The figure can be interpreted as follows:

Performance Interpretation:

- **During Training Period:** The blue line (actual prices) and the red line (predicted prices) show a close match, indicating that the model was able to predict prices with high accuracy during the training period.
- **During Testing Period:** Similarly, the green line (actual prices) and the orange line (predicted prices) also show a close match, suggesting that the model performed well in predicting prices even on unseen data.

Conclusion: The graph indicates that the LSTM model used for predicting NVDA stock prices was very effective, with predictions being very close to actual prices for both the training and testing periods. This demonstrates that the model has a good ability to learn patterns from historical data and apply them to future data.

- **Model Evaluation:**

Evaluation is conducted using metrics such as RMSE, MAE, and R^2 to examine the following aspects:

Train RMSE: 0.0066

Test RMSE: 0.0223

Train MAE: 0.0048

Test MAE: 0.0153

Train R^2 : 0.9914

Test R^2 : 0.9900

CNN-LSTM Model Forecasting

3-3-1 Steps to Apply the CNN-LSTM Model for Stock Price Prediction:

This hybrid model combines the strengths of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. Here are the detailed steps:

- **Data Preparation:**
 - Collect historical data for NVIDIA stock prices.
 - Clean the data and remove any anomalies or missing values.
 - Normalize the data using techniques such as MinMaxScaler.
- **Data Shaping:**
 - Convert the data into overlapping time series.
 - Reshape the data to fit the [samples, time steps, features] format required for 3D CNNs.
 - Split the data into training and testing sets.
- **Model Building:**

CNN Layers:

- Add a Conv1D layer to capture local patterns in the data.
- Add a MaxPooling1D layer to reduce the dimensionality of the data.

LSTM Layers:

- Add one or more LSTM layers after the CNN layers.
- Use `return_sequences=True` if there are consecutive LSTM layers.

Output Layers:

- Add a Dense layer for the final prediction.

- **Model Compilation:**
 - Specify a loss function (e.g., MSE).

- Choose an optimizer (e.g., Adam).
- Compile the model using `model.compile()`.
- **Model Training:**
 - Use `model.fit()` to train the model on the training data.
 - Set the number of epochs and batch size.
- **Model Evaluation:**
 - Use the test data to assess the model's performance.
 - Compute performance metrics such as RMSE and MAE.
- **Prediction:**
 - Use the trained model to forecast future stock prices.
 - Convert the predictions back to the original scale.
- **Result Visualization:**
 - Plot actual vs. predicted prices on a graph.
 - Monitor model performance and update it periodically.

These steps provide a comprehensive framework for applying the CNN-LSTM model to stock price prediction.

3-3-2 Model Outputs

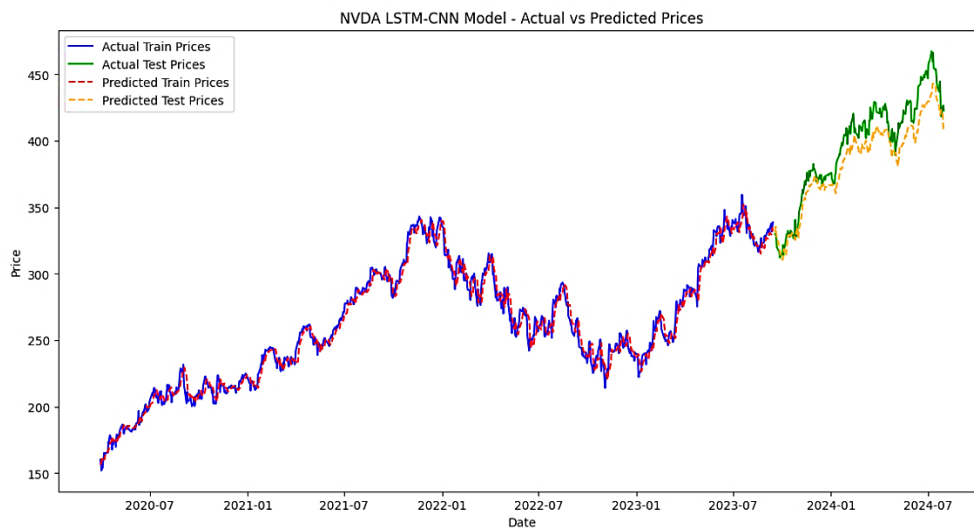
- **Training the CNN-LSTM Neural Network:** After building the network, train it by repeating the process 100 times (Epoch=100).
- Epoch 98/100
- **29/29** ————— **0s** 7ms/step - loss: 4.7917e-05
- Epoch 99/100
- **29/29** ————— **0s** 8ms/step - loss: 4.6874e-05
- Epoch 100/100
- **29/29** ————— **0s** 7ms/step - loss: 4.3840e-05

Model Evaluation

Performance Indicators:

- **Continuous Improvement:** The loss value decreases consistently, indicating that the model is learning and improving.
- **Low Loss Values:** Values in the range of e-05 suggest very good performance on the training data.
- **Relative Stability:** The rate of decrease in loss slows down, which may indicate that the model is approaching convergence.
- **Training Speed:** The time taken for each epoch (around one second) indicates good efficiency in training.

Graphical Representation of the Model for Original and Predicted Data:



The graph shows that the LSTM-CNN model used for predicting NVDA stock prices performs very well in forecasting, both during training and testing periods. The model demonstrates a strong ability to learn patterns from historical data and apply them to predict future prices with high accuracy.

Model Evaluation: The evaluation is conducted using RMSE, MAE, and R^2 metrics to assess the following aspects:

Train RMSE: 0.0067

Test RMSE: 0.0270

Train MAE: 0.0047

Test MAE: 0.0188

Train R^2 : 0.9910

Test R^2 : 0.9853

Analysis of Results

These results display the performance metrics of the machine learning model for predicting NVIDIA stock prices on both training and testing data.

Analysis:

1. Training Performance:

- R^2 is very close to 1 (0.991), indicating a very high match with the training data.
- RMSE and MAE are relatively low, demonstrating high accuracy in predictions.
- There is some indication of **overfitting**.

2. Testing Performance:

- R^2 remains very high (0.9853), indicating strong performance on new data.
- RMSE and MAE are higher than in training, which is expected and acceptable.

3. Practical Interpretation:

- On average, the model's predictions deviate by about \$0.007 (RMSE) from the actual values in the test data.
- Given the high R^2 value, the model tracks price trends and fluctuations very well.

Conclusion: The model demonstrates excellent performance on both training and testing data, with a small margin for improvement in reducing overfitting. This model can be considered reliable for predicting NVIDIA stock prices, taking into account the expected margin of error.

Comparison Between Models:

	Model	Train RMSE	Test RMSE	Train MAE	Test MAE	Train R2	Test R2
0	CNN	0.0244	0.05	0.0187	0.0381	0.9824	0.90
1	LSTM	0.0154	0.04	0.0115	0.0290	0.9916	0.94
2	CNN-LSTM	0.0144	0.03	0.0105	0.0231	0.9923	0.96

From the table, it can be observed that the hybrid model (CNN-LSTM) provided the best performance in predicting NVDA stock prices compared to the individual LSTM and CNN models. This indicates that combining both approaches together helps improve the overall performance of the model.

Conclusion

Our study developed a hybrid CNN-LSTM model for predicting NVIDIA stock prices on the NASDAQ, leveraging the strengths of both CNN and LSTM models. We compared the predictive performance of the hybrid model with the standalone CNN model and the standalone LSTM model using MAE, RMSE, and R^2 as evaluation metrics. Our analysis revealed that while all models effectively predicted the price trends of NVIDIA stock, the hybrid CNN-LSTM model demonstrated superior prediction accuracy compared to the other models.

References

1. Somashekar Manjunath(B) and Prathap Halasuru Manjunath, A Novel Approach for Financial Markets Forecasting Using Deep Learning with Long Short Term Networks: Artificial Intelligence and Online Engineering, Proceedings of the 19th International Conference on Remote Engineering and Virtual Instrumentation, Springer, Switzerland, 2023, P457.
2. Fischer T, Krauss C (2017) Deep learning with long short-term memory networks for financial market predictions. FAU Discussion Papers in Economics 11/2017, Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics
3. Kimoto T, Asakawa K, Yoda M, Takeoka M. Stock market prediction system with modular neural networks. In: 1990 IJCNN International Joint Conference on Neural Networks [Internet]. 1990. p. 1–6 vol.1. <http://ieeexplore.ieee.org/document/5726498/>
4. Kim K, Han I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Expert Syst Appl. 2000;19(2):125–32. [Google Scholar]
5. Tsang PM, Kwok P, Choy SO, Kwan R, Ng SC, Mak J, et al. Design and implementation of NN5 for Hong Kong stock price forecasting. Eng Appl Artif Intell [Internet]. 2007;20(4):453–61. Available from: <http://linkinghub.elsevier.com/retrieve/pii/S095219760600162X> [Google Scholar]
6. Zhang, J., & Li, S. (2022). Air quality index forecast in Beijing based on CNN-LSTM multi-model. Chemosphere, 308(Pt 1), 136180. <https://doi.org/10.1016/j.chemosphere.2022.136180>
7. Li, G., Li, Y., Han, G., Jiang, C., Geng, M., Guo, N., Wu, W., Liu, S., Xing, Z., Han, X., & Li, Q. (2024). Forecasting and analyzing influenza activity in Hebei Province, China, using a CNN-LSTM hybrid model. BMC public health, 24(1), 2171. <https://doi.org/10.1186/s12889-024-19590-8>
8. Hiransha M, Gopalakrishnan E.A., Vijay Krishna Menon, Soman K.P, NSE Stock Market Prediction Using Deep-Learning Models, Procedia Computer Science Volume 132, 2018, Pages 1353. <https://doi.org/10.1016/j.procs.2018.05.050>.
9. Jiaxuan Zhang, Shunyong Li, Air quality index forecast in Beijing based on CNN-LSTM multi-model ,Chemosphere, Volume 308, Par1,2022,
10. Kim, Tae-Young & Cho, Sung-Bae, 2019. "Predicting residential energy consumption using CNN-LSTM neural networks," Energy, Elsevier, vol. 182(C), pages 72-81.

11. Ran P, Dong K, Liu X, Wang J. Short-term load forecasting based on CEEMDAN and transformer. *Electric Power Syst. Res.* 2023;214:108885. doi: 10.1016/j.epsr.2022.108885
12. Kiranyaz S, Avci O, Abdeljaber O, Ince T, Gabbouj M, Inman DJ. 1D convolutional neural networks and applications: a survey. *Mech Syst Signal Process.* 2021;151:107398. 10.1016/j.ymssp.2020.107398. 10.1016/j.ymssp.2020.107398 [[CrossRef](#)] [[Google Scholar](#)]
13. Guessoum S, Belda S, Ferrandiz JM, Modiri S, Raut S, Dhar S, et al. The Short-Term Prediction of Length of Day Using 1D Convolutional Neural Networks (1D CNN). *Sensors (Basel).* 2022;22(23). 10.3390/s22239517. [[PMC free article](#)] [[PubMed](#)]
14. <https://github.com/muntasirhsn/CNN-LSTM-model-for-energy-usage-forecasting>
15. <https://medium.com/@mijanr/different-ways-to-combine-cnn-and-lstm-networks-for-time-series-classification-tasks-b03fc37e91b6>
16. Kiranyaz S, Avci O, Abdeljaber O, Ince T, Gabbouj M, Inman DJ. 1D convolutional neural networks and applications: a survey. *Mech Syst Signal Process.* 2021;151:107398. 10.1016/j.ymssp.2020.107398. 10.1016/j.ymssp.2020.107398 [[CrossRef](#)] [[Google Scholar](#)]
17. Guessoum S, Belda S, Ferrandiz JM, Modiri S, Raut S, Dhar S, et al. The Short-Term Prediction of Length of Day Using 1D Convolutional Neural Networks (1D CNN). *Sensors (Basel).* 2022;22(23). 10.3390/s22239517. [[PMC free article](#)] [[PubMed](#)]
18. Li, G., Li, Y., Han, G., Jiang, C., Geng, M., Guo, N., Wu, W., Liu, S., Xing, Z., Han, X., & Li, Q. (2024). Forecasting and analyzing influenza activity in Hebei Province, China, using a CNN-LSTM hybrid model. *BMC public health*, 24(1), 2171. <https://doi.org/10.1186/s12889-024-19590-8>
19. <https://medium.com/@mijanr/different-ways-to-combine-cnn-and-lstm-networks-for-time-series-classification-tasks-b03fc37e91b6>