## 1. LSTM Architecture

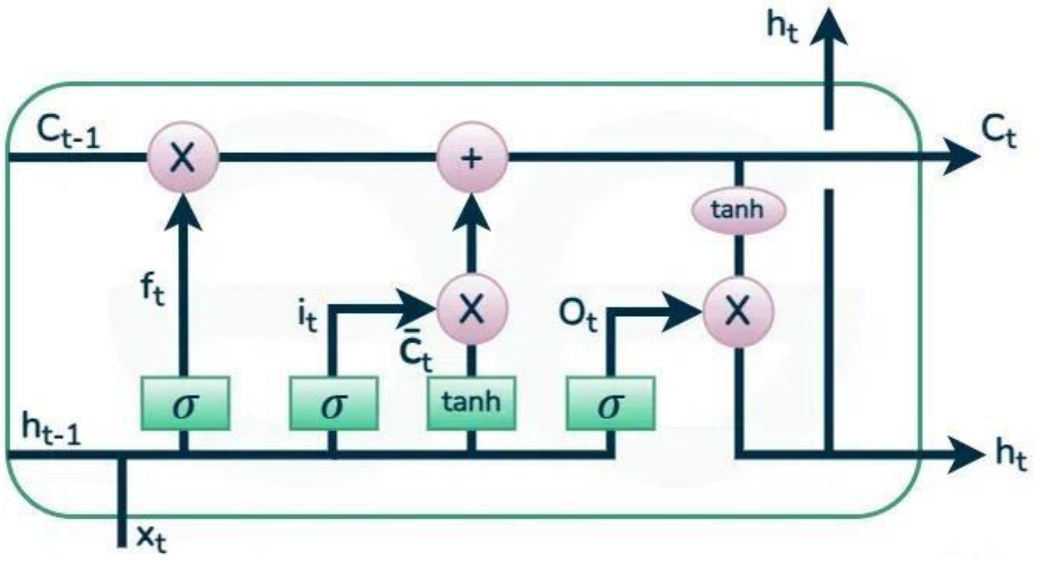
LSTM architectures involves the memory cell which is controlled by three gates: the *input gate*, the *forget gate* and the *output gate*. These gates decide what information to add to, remove from and output from the memory cell.

- **Input gate**: Controls what information is added to the memory cell.
- **Forget gate**: Determines what information is removed from the memory cell.
- **Output gate**: Controls what information is output from the memory cell.

This allows LSTM networks to selectively retain or discard information as it flows through the network which allows them to learn long-term dependencies. The network has a hidden state which is like its short-term memory. This memory is updated using the current input, the previous hidden state and the current state of the memory cell.

## 2. Working of LSTM

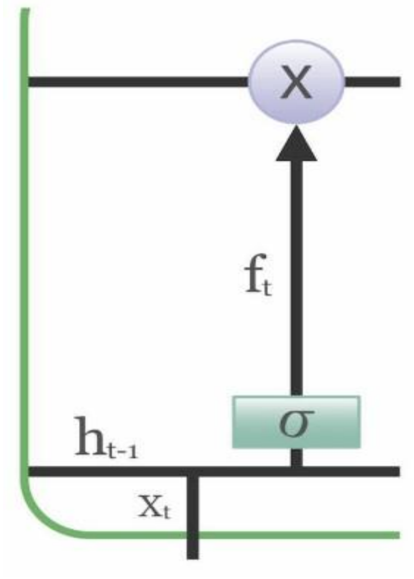
LSTM architecture has a chain structure that contains four neural networks and different memory blocks called **cells**.



Information is retained by the cells and the memory manipulations are done by the **gates.** There are three gates –

### Forget Gate

The information that is no longer useful in the cell state is removed with the forget gate. Two inputs *$x_t$* (input at the particular time) and *$h_{t-1}$* (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.



The equation for the forget gate is:

$$f_t=\sigma(W_f\cdot[h_{t-1},x_t]+b_f)$$

where:

- W_f represents the weight matrix associated with the forget gate.
- [h_t-1, x_t] denotes the concatenation of the current input and the previous hidden state.
- B_f is the bias with the forget gate.
- σ is the sigmoid activation function.

## Input Gate

The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs *$h_{t-1}$* and *$x_t$*. . Then, a vector is created using *tanh* function that gives an output from -1 to +1, which contains all the possible values from $h_{t-1}$ and *$x_t$*. At last, the values of the vector and the regulated values are multiplied to obtain the useful information. The equation for the input gate is:

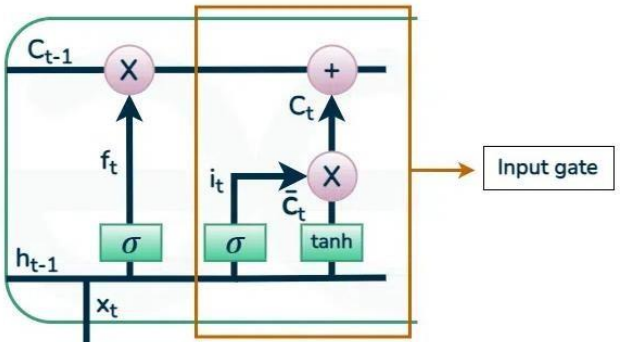
$$i_t = \sigma(W_i\cdot [h_{t-1}, x_t] + b_i)$$

$$\hat{C}_t = tanh(W_c\cdot [h_{t-1}, x_t] + b_c)$$

We multiply the previous state by $f_t$, disregarding the information we had previously chosen to ignore. Next, we include $i_t * C_t$. This represents the updated candidate values, adjusted for the amount that we chose to update each state value.

$C_t = f_t \odot C_{t-1} + i_t \odot C_t$ Where

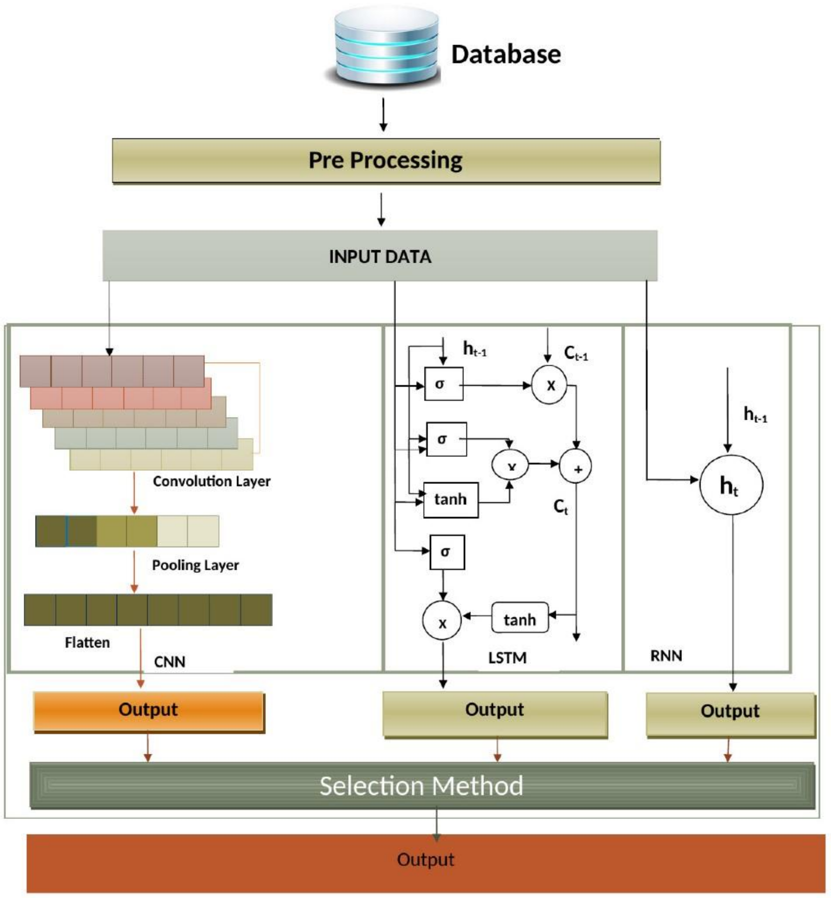
- $\odot$ denotes element-wise multiplication
- tanh is tanh activation function



## Output Gate

The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying tanh function on the cell. Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs ht−1$h_{t-1}$and xt$x_t$. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell. The equation for the output gate is:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$



## 3. Mathematical Applications of LSTM

### 1. Sequential Data Modeling

LSTMs are particularly effective for modeling sequential data, where the order of inputs matters. This includes:

- **Time-series forecasting**: Predicting future values based on past observations (e.g., stock prices, weather data).
- **Natural Language Processing (NLP)**: Processing text data, where the sequence of words or characters is critical (e.g., machine translation, text generation).
- **Speech recognition**: Modeling audio signals as sequential data.

**Mathematical Representation**:
LSTMs process sequential data by maintaining a hidden state ht*ht* at each time step t*t*, which captures information from previous time steps. The hidden state is updated iteratively as new inputs are processed.

## 2. Backpropagation Through Time (BPTT)

LSTMs are trained using a variant of backpropagation called Backpropagation Through Time (BPTT). This involves:

- Unfolding the LSTM over time steps.
- Computing gradients with respect to the loss function.
- Updating the weights using gradient descent.

The mathematical challenge lies in handling the vanishing/exploding gradient problem, which LSTMs mitigate through their gating mechanisms.

## 3. Applications in Dynamical Systems

LSTMs are used to model and predict the behavior of dynamical systems, such as:

- **Chaotic systems**: Predicting trajectories in systems like the Lorenz attractor.
- **Control systems**: Modeling and controlling systems with timedependent inputs.

**Mathematical Formulation**:
LSTMs can approximate the state transition function f*f* of a dynamical system:

$$\hat{x}_{t+1} = f(x_t, u_t)$$

where xt*xt* is the state and ut*ut* is the input at time t*t*.

## 4. Sequence-to-Sequence Mapping

LSTMs are used for tasks that involve mapping one sequence to another, such as:

- **Machine translation**: Mapping a sequence of words in one language to another.
- **Speech-to-text**: Converting a sequence of audio signals to text.

**Mathematical Representation**:
The encoder-decoder architecture uses two LSTMs:

- The encoder LSTM processes the input sequence and encodes it into a fixed-size context vector.
- The decoder LSTM generates the output sequence based on the context vector.

5. **Probability Estimation**

LSTMs can be used to model probability distributions over sequences, such as:

- **Language modeling**: Estimating the probability of the next word in a sequence.
- **Sequence generation**: Sampling sequences from a learned distribution.

**Mathematical Formulation**:
The output of the LSTM is often passed through a softmax function to produce a probability distribution:

$$P(y_t|y_{<t}, x) = \text{softmax}(W_y \cdot h_t + b_y)$$

where yt*yt* is the predicted output at time t*t*, and ht*ht* is the hidden state.

6. **Handling Irregular Time Steps**

LSTMs can be adapted to handle irregularly sampled time-series data by incorporating time intervals into the model. This is useful in applications like:

- **Medical data analysis**: Modeling patient records with irregular measurements.
- **Event-based data**: Processing data where events occur at irregular intervals.

**Mathematical Adaptation**:
The time interval Δt*Δt* can be included as an additional input:

$$h_t = \text{LSTM}(x_t, \Delta t, h_{t-1})$$

7. **Matrix Operations and Parallelization**

LSTM computations involve large matrix multiplications and element-wise operations, which can be efficiently parallelized on GPUs. This makes them scalable for large datasets.

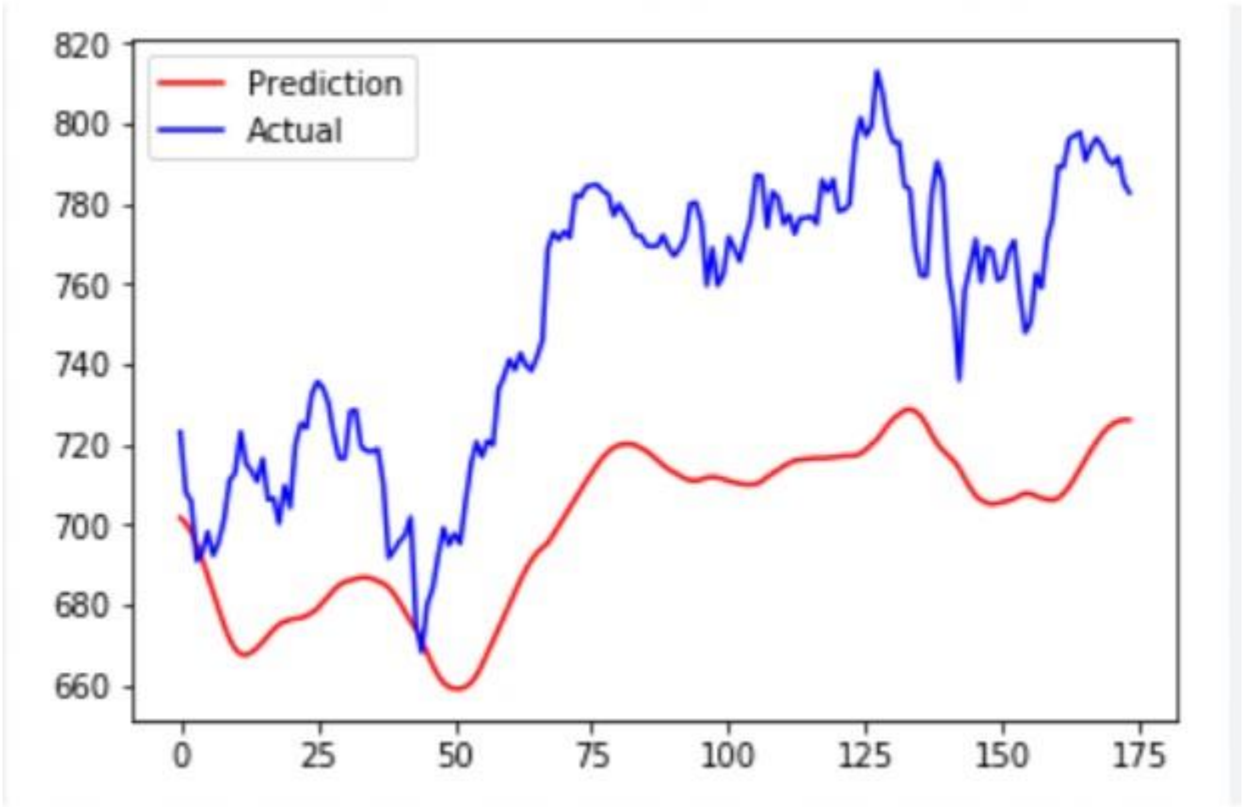
**Key Operations**:

- Matrix-vector multiplications for gate computations.
- Element-wise operations (e.g., sigmoid, tanh, Hadamard product).\

8. **Extensions and Variants**

LSTMs have inspired several variants, such as:

- **Gated Recurrent Units (GRUs)**: A simplified version of LSTMs with fewer parameters.
- **Bidirectional LSTMs**: Process sequences in both forward and backward directions.


Prediction
Actual