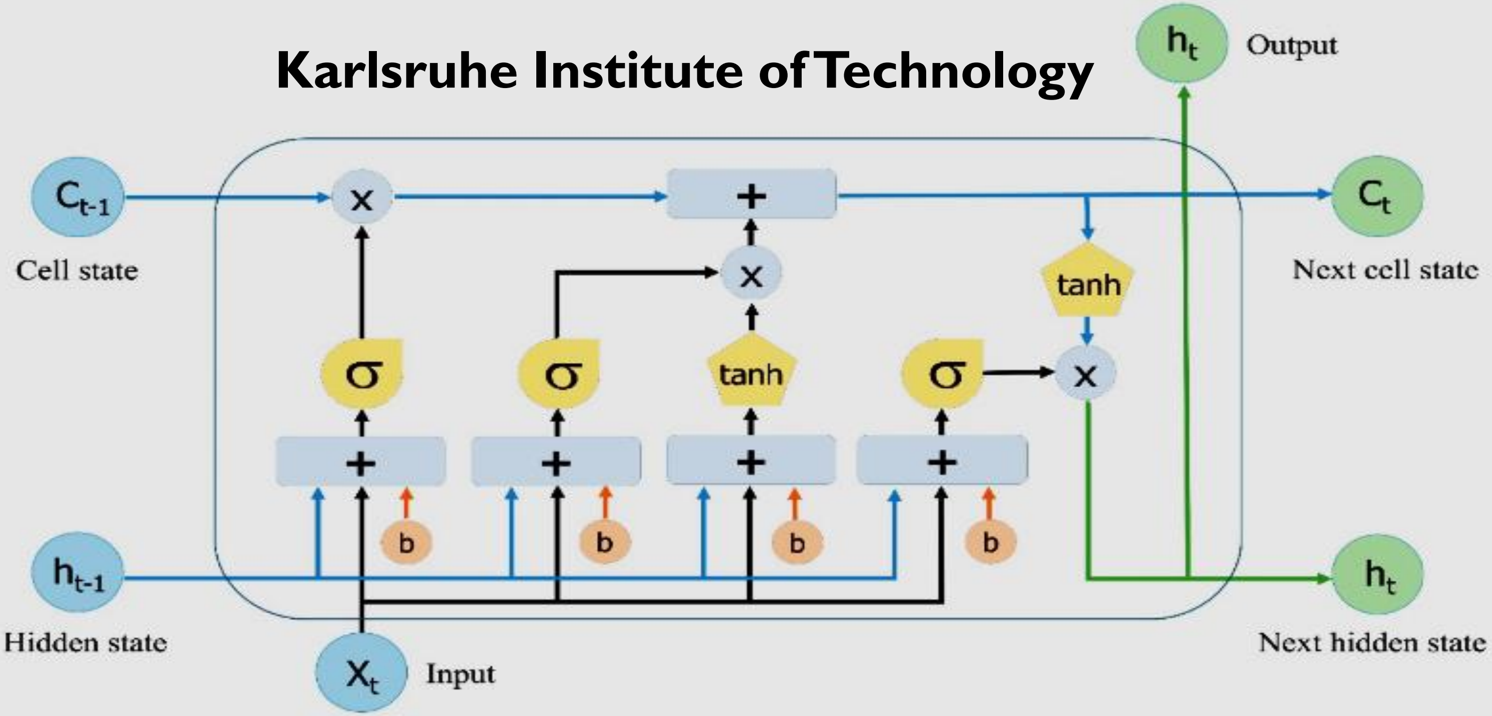
# Music Generation with LSTM Networks
# Date: 06 May, 2025

**<u>Supervisors: Brojogopal Sapui, Priyanjana Pal</u>**

**Karlsruhe Institute of Technology**



# Outline

1 Motivation

2 Introduction to LSTM

3 Music Generation with LSTM

4 Project Workflow

5 Tools and Timeline

# Why Music Generation with LSTM Networks?

- Traditional music composition methods are labor-intensive and limited by human creativity.

- Machine Learning offers the ability to explore and generate diverse, novel musical ideas.

- RNNs can model sequential data effectively but struggle with long-term dependencies.

- LSTMs address this issue, allowing networks to remember and utilize important information from earlier in the sequence.

# Introduction to LSTM

- **What is LSTM?**
  - Long Short-Term Memory (LSTM) is a special kind of RNN architecture.

  - Capable of learning long-range dependencies in sequence prediction tasks.

  - Consists of gates (input, forget, output) which control the flow of information, enhancing the memory capacity over traditional RNNs.

# Introduction to Music Generation

Karlsruher Institut für Technologie

- **What is Music Generation?**
  - Automated creation of musical pieces using computational models.
  - Utilizes computational algorithms to emulate the creative processes of human composers.
  - Generates original musical sequences or extensions of existing pieces.

- **Why use Machine Learning?**
  - Enables analysis of large datasets to recognize complex patterns within musical data.
  - Facilitates automated learning and creativity without explicitly programming rules.
  - Allows the generation of novel and diverse musical compositions that maintain coherent structure and musicality.
  - Offers scalability and continuous improvement as more data becomes available.

# Basics to Know

- **Machine Learning Fundamentals:**
  - Supervised vs. Unsupervised learning
  - Sequence modeling and prediction
- **Deep Learning Basics:**
  - Neural Networks
  - Recurrent Neural Networks (RNN)

# What is LSTM?

- It is a special type of RNN, capable of learning long-term dependencies.
- Long Short Term Memory (LSTM) is a type of deep learning model that is mostly used for analysis of sequential data (time series data prediction).
- There are different application areas that are used:
  - Language model,
  - Neural machine translation,
  - Music generation,
  - Time series prediction,
  - Financial prediction,
  - Robot control,
  - Time series prediction,
  - Speech recognition,
  - Rhythm learning, Music composition, Grammar learning, Handwriting recognition etc

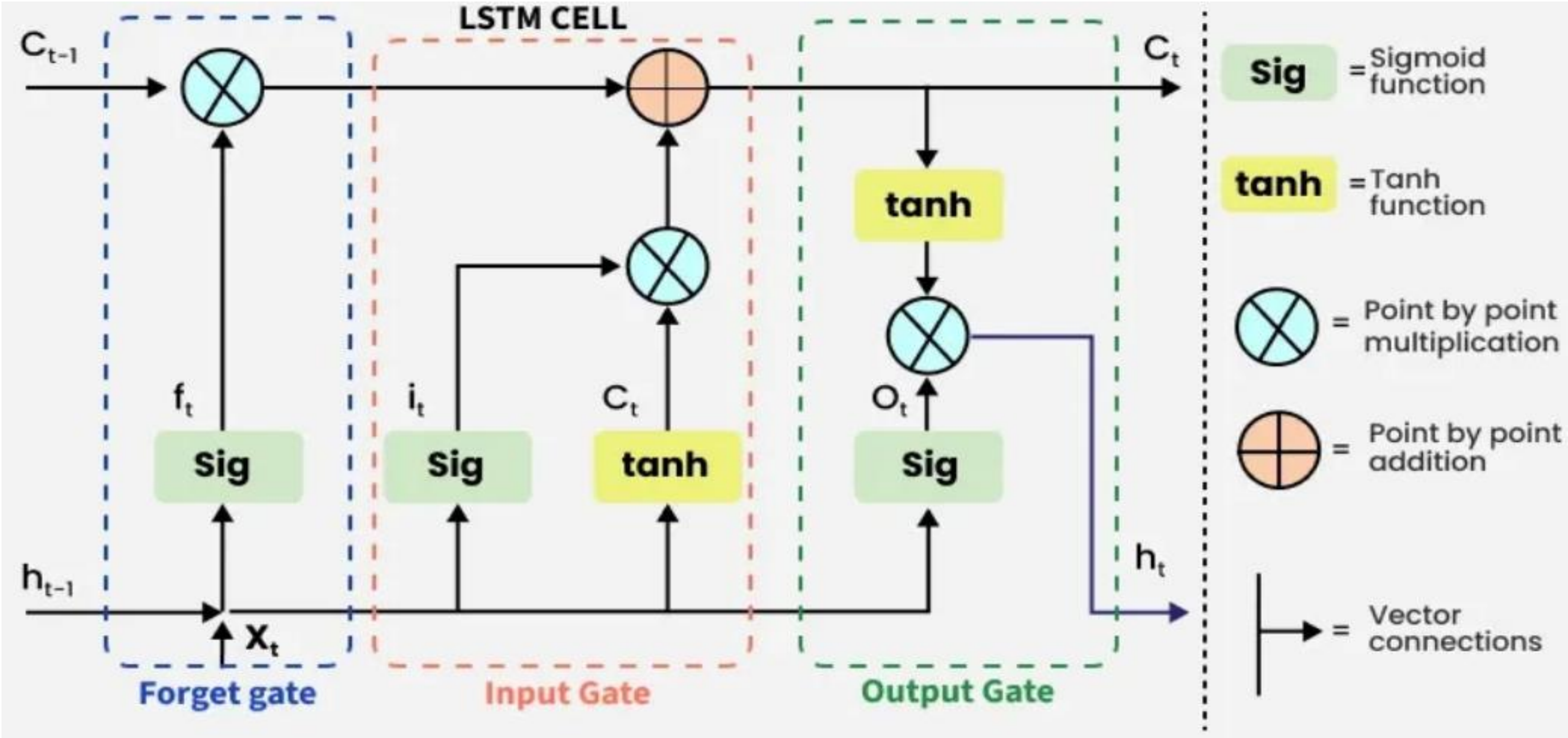
# Essential Concepts of LSTM



- Gates in LSTM:
  - 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.
- Role of gates: Manage information flow and memory in sequential data

# 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 –




https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

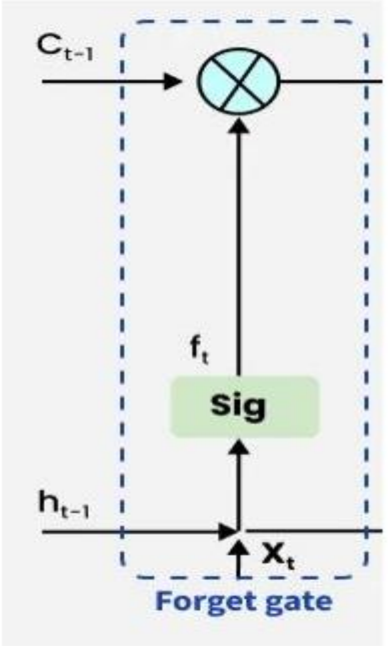
# Working of LSTM

- **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 concatenation of the current input & previous hidden state.
- $b_f$ is the bias with the forget gate., σ is the sigmoid activation function.



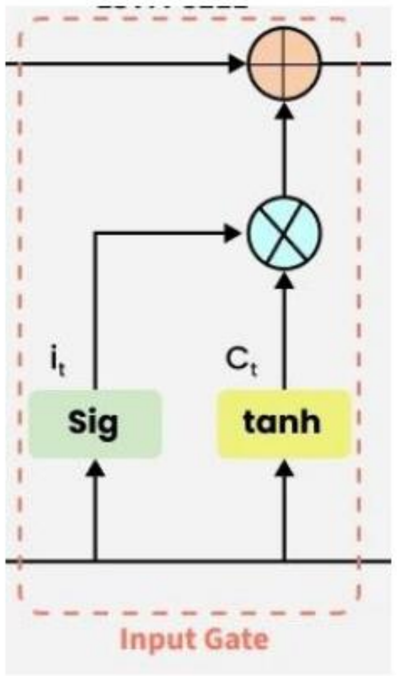

Forget Gate

# Working of LSTM

- **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, values of the vector and the regulated values are multiplied to

  obtain the useful information. The equation for the input gate is:
  - $i_t = \sigma(W_i \cdot [h_t - 1, x_t] + b_i)$

$$C_t = \tanh(W_c \cdot [h_t - 1, x_t] + b_c)$$



*Input Gate*

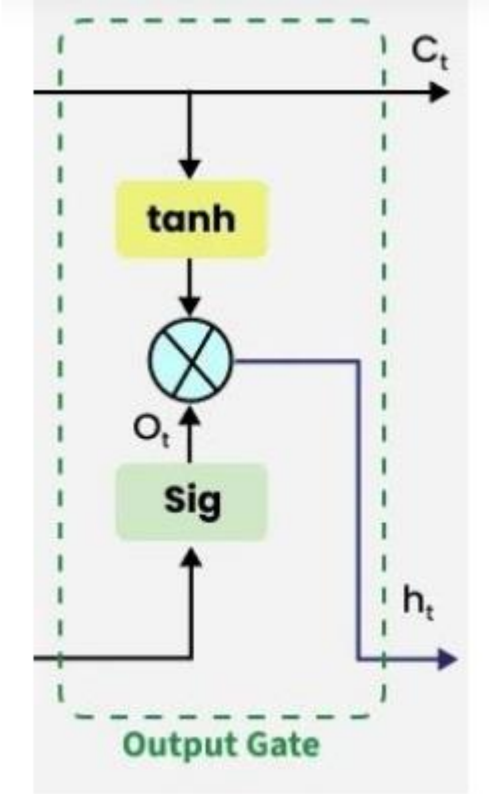
We multiply the previous state by $f_t$, disregarding the information we had previously chosen to ignore. Next, we include $i_t * C_t$. This represents the upd candidate values, adjusted that we chose to update each state value.

$$C_t = f_t \odot C_{t\,\_\,1} + i_t \odot C^\wedge{}_t$$

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

# Working of LSTM

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



Output Gate

# File Types and Data Preparation

Karlsruher Institut für Technologie

- **MIDI Files:**
  - Primary format for music generation
  - Contains information on notes, duration, tempo, instruments
- **Data Preprocessing Steps:**
  - Conversion of MIDI files into sequences understandable by the neural network
  - Tokenization and encoding of musical data

# Project Workflow



- **Data Collection:** Gather MIDI files from diverse sources.
- **Preprocessing:** Convert MIDI files into suitable numerical sequences.
- **Model Training:** Utilize LSTM to learn sequential patterns.
- **Generation:** Produce new musical sequences based on learned patterns.
- **Evaluation:** Assess generated music qualitatively (listening) and quantitatively (pattern similarity).

# Tools and Technologies

- **Programming Language**: Python
  - Libraries and Frameworks: TensorFlow/Keras for neural network implementation
  - Music21 or pretty_midi for MIDI file handling
- **IDE and Tools**
  - Jupyter Notebooks
  - GitHub for version control

References:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/
https://github.com/Kulbear/deep-learning-coursera/tree/master/Sequence%20Models
https://www.kaggle.com/code/pablocastilla/predict-stock-prices-with-lstm/notebook

# Project Organization Suggestions

Karlsruher Institut für Technologie

- GitHub Repository Structure:
  - /data: Raw and processed MIDI files
  - /src: Python scripts for preprocessing, training, generation
  - /models: Saved trained LSTM models
  - /outputs: Generated music files (MIDI/WAV)
  - README.md: Clearly document steps for project setup and execution

# Milestones & Timeline

- **Week 1-2:** Basic concepts and tools familiarization
- **Week 3-4:** Data collection and preprocessing
- **Week 5-7:** Initial model training
- **Week 7-9:** Music generation tests
- **Week 10-12:** Project refinement and documentation and presentation (by September 15)

# Thank you for your attention!

Brojogopal Sapui (brojogopal.sapui@kit.edu)
Priyanjana Pal (priyanjana.pal@kit.edu)