CLASSIFYING HUMAN ACTIVITIES USING THE LRCN MODEL

Presented by

Hoang Thi Hoai Thuong

Luong Nguyen Minh Chanh

Overview

01 Introduction

02 Dataset

03 LSTM

04 Model Explaination

05 Preprocess on Video

06 Training Model

07 Demo

08 Conclusion

Classifying Human Activities

- Human activity classification involves identifying and categorizing various physical activities performed by individuals based on collected data.

- In this project, we focus on video data to classify activities such as walking, running, sitting, standing, jumping, and more.

Diverse Action Types

A well-known data set in the field of action recognition, which consists of 50 different types of actions. These actions range from everyday activities to complex sports activities. The diversity of action types makes it an interesting challenge for action recognition systems.

Data Size and Richness

Consists of over 6000 short videos, each containing different actions in a variety of contexts and lighting conditions. This provides a rich data source for training and testing deep learning models, ensuring that the system can handle a wide range of real-world scenarios.

Widely used in deep learning research

Provides a powerful platform for testing and comparing the performance of different models.
Helps to accelerate the research progress in the field of artificial intelligence and deep learning.

Data Preprocessing Consistency

The videos in UCF50 are preprocessed to normalize the frame size to 64x64 pixels and fix the frame sequence length to 20. This preprocessing not only enhances the data consistency but also optimizes the model learning ability, ensuring that important features are extracted efficiently.

Explore Temporal Data

Sequential Nature of Activities

Human activities consist of a series of movements that occur in a specific order. Temporal data captures this sequence, which is essential for distinguishing between different activities.

Duration and Timing

The duration of each movement and the timing between movements are important features. For example, the time intervals between steps can help differentiate between walking and running.

Contextual Information

Temporal data provides context. A single frame or a short segment might not provide enough information to classify an activity accurately. However, a sequence of frames or sensor readings can provide the necessary context.

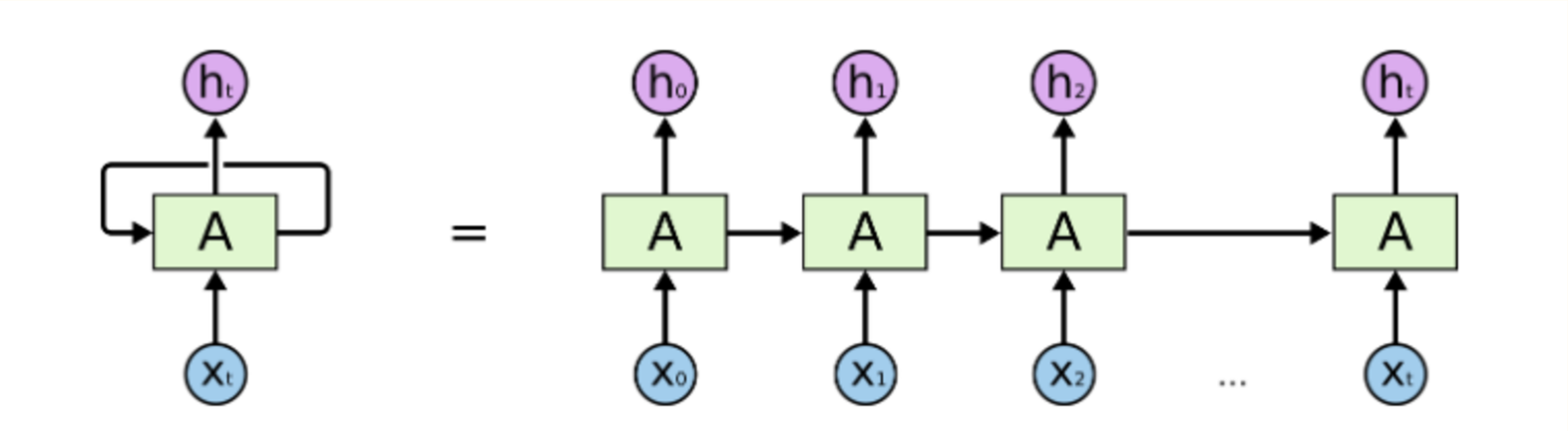
Dynamic Changes

Activities involve dynamic changes over time. Temporal data captures these changes, allowing the model to understand the progression of movements.

Why RNNs are Used for Temporal Feature Extraction?

Recurrent Neural Networks (RNNs) are specifically designed to handle sequential data and capture temporal dependencies. Here are the key reasons why RNNs are suitable for temporal feature extraction in human activity classification:

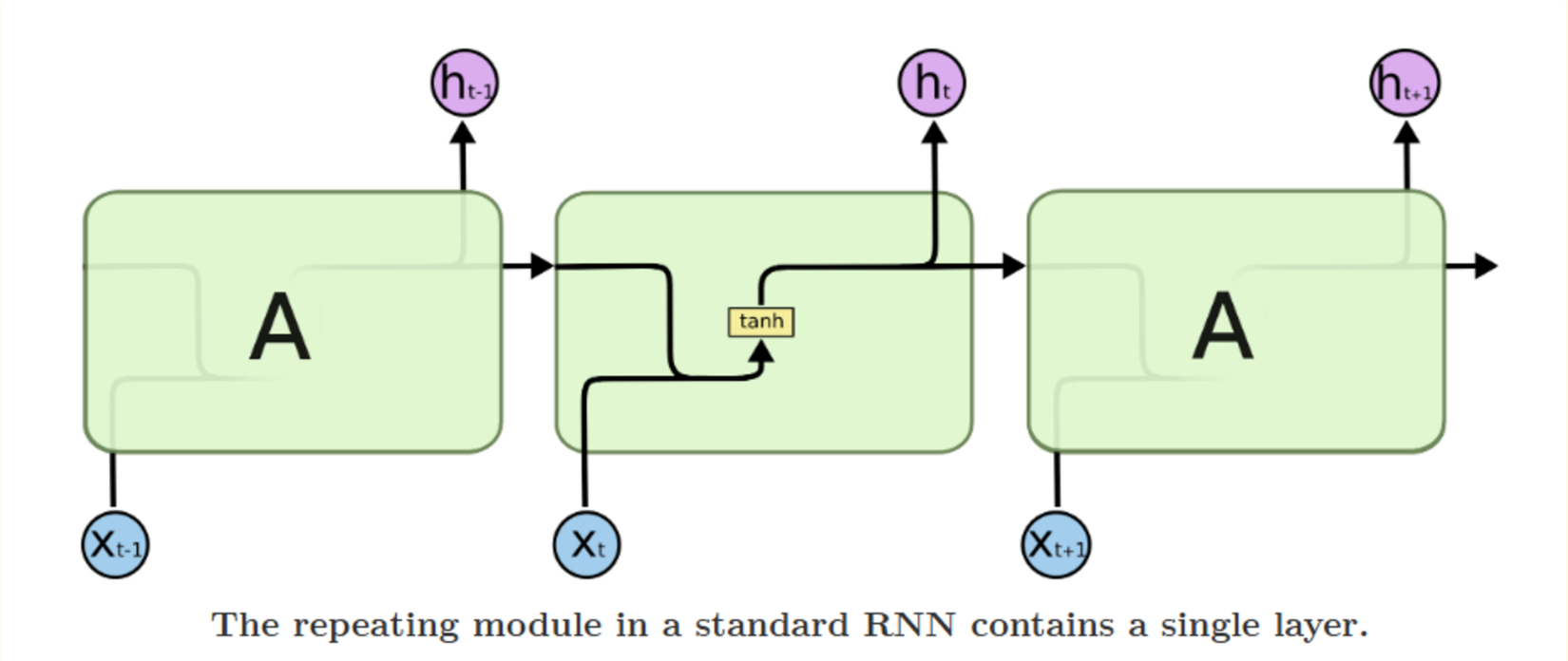
- **Memory Mechanism**: Retains information from previous time steps.
- **Sequential Processing**: Processes data in order, capturing temporal dependencies.



Problem of Long-Term Dependencies in Traditional RNNs

Traditional Recurrent Neural Networks (RNNs) face significant challenges when it comes to learning long-term dependencies in sequential data. This problem is primarily due to the following issues:

- Vanishing Gradient Problem: Gradients become very small during backpropagation, making it hard to learn long-term dependencies.
- Short-Term Memory: Traditional RNNs focus on short-term dependencies due to the vanishing gradient problem, limiting their ability to capture long-term dependencies.



The repeating module in a standard RNN contains a single layer.

Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) architecture designed to effectively capture long-term dependencies in sequential data. LSTMs address the limitations of traditional RNNs, such as the vanishing and exploding gradient problems, by incorporating a more complex cell structure.

How LSTM Provides Long-Term Memory

LSTMs achieve long-term memory through their unique cell structure, called LSTM cells.

LSTM Cell

Cell State

The cell state is the memory of the LSTM cell, which carries information across different time steps. It can be thought of as a conveyor belt that runs through the entire sequence, with some minor linear interactions.

Hidden State

The hidden state is the output of the LSTM cell at each time step. It is used for making predictions and is also passed to the next time step.

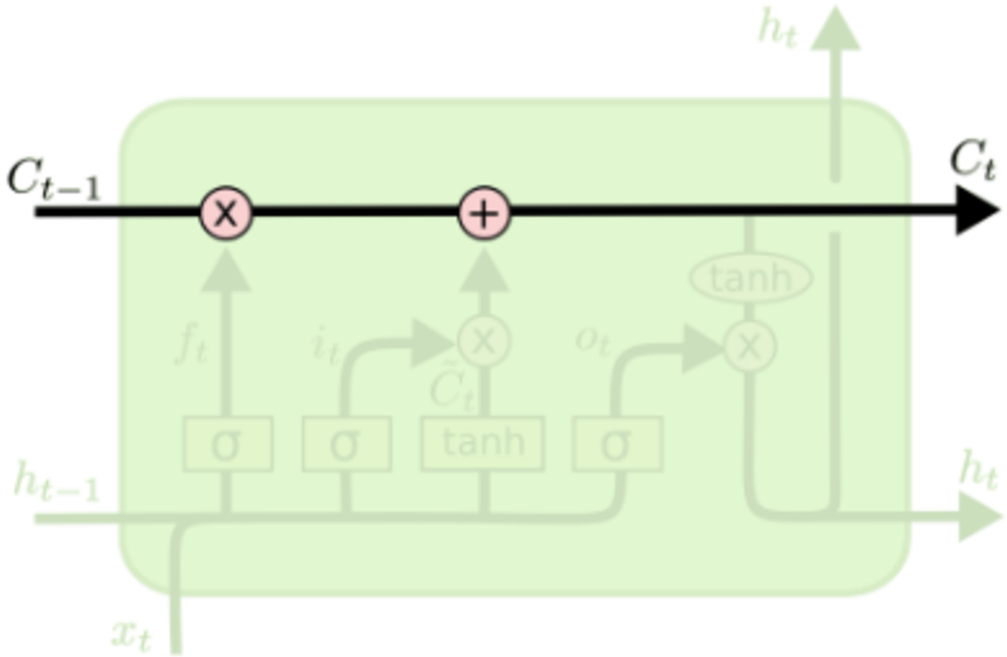
Gates

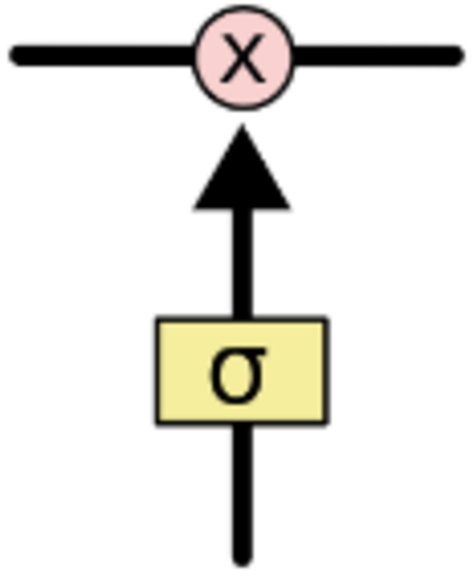
3 Gate Types

- **Forget Gate**: Decides what information to discard from the cell state.
- **Input Gate**: Decides which new information to add to the cell state.
- **Output Gate**: Decides what part of the cell state to output as the hidden state.

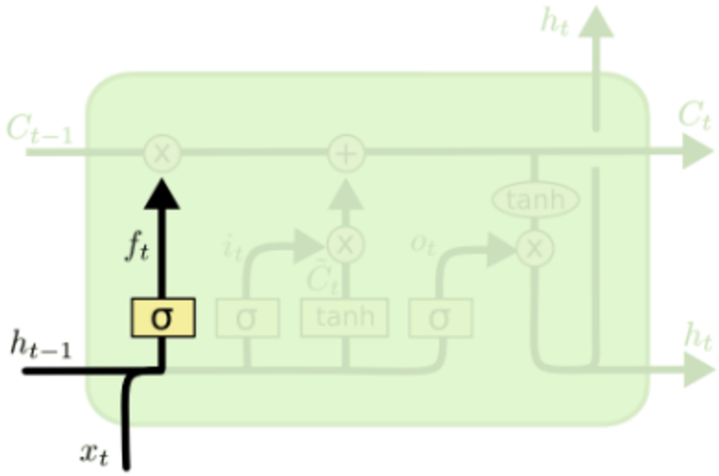
Components

How LSTMs Remember Information



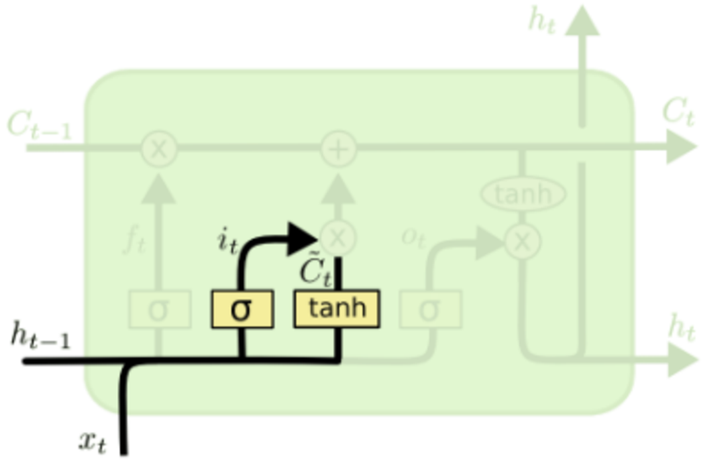


How LSTMs Remember Information



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

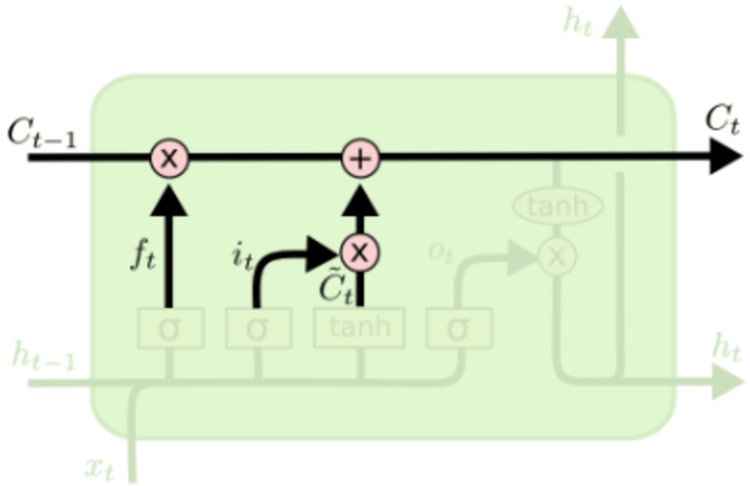
How LSTMs Remember Information



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

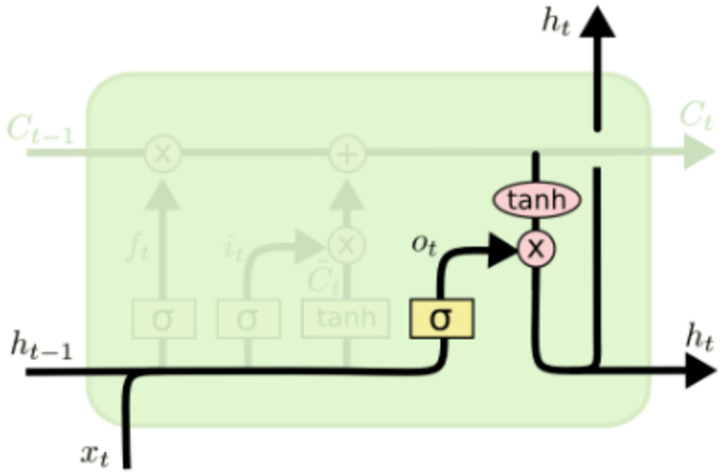
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

How LSTMs Remember Information



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

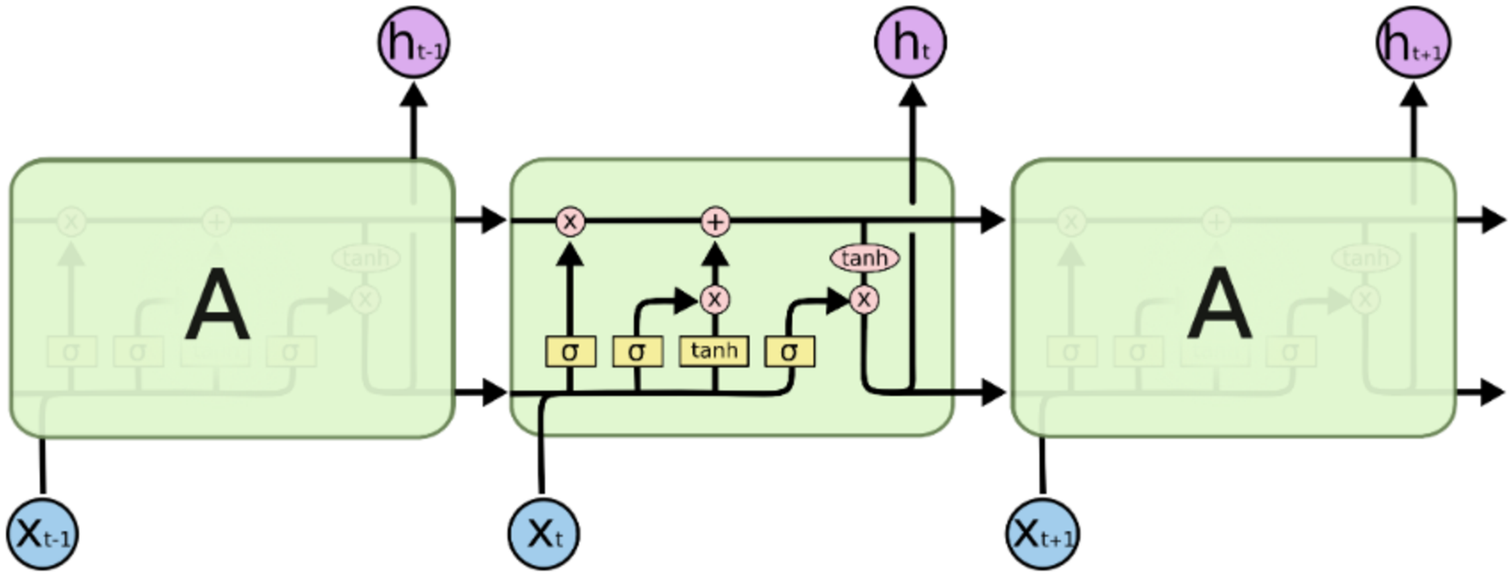
How LSTMs Remember Information



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

$$h_t = o_t * \tanh\left(C_t\right)$$

How LSTMs Remember Information



Long-term Recurrent Convolutional Networks (LRCNs)

Long-term Recurrent Convolutional Networks (LRCNs) are a type of deep learning model that combines Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), specifically LSTMs, to handle tasks that involve both spatial and temporal information. LRCNs are particularly effective for tasks such as video classification, activity recognition, and sequence prediction.

Key Components of LRCN Models:

Convolutional Neural Networks (CNNs)

- Extract spatial features from the input data.
- CNNs are used to process and extract features from images or frames in a video. They capture spatial hierarchies and patterns, such as edges, textures, and objects.
- In video classification, CNNs can be used to extract features from each frame of the video.

Recurrent Neural Networks (RNNs) with LSTM Cells

- Extract temporal features and capture dependencies over time.
- LSTMs are used to process the sequence of spatial features extracted by the CNN. They capture the temporal dynamics and dependencies between consecutive frames or time steps.
- In video classification, LSTMs can be used to analyze the sequence of features extracted from each frame to understand the temporal context and classify the activity.

Model Initialization

Initializes a sequential model, which is a linear stack of layers.

Time Distributed Layer

The TimeDistributed wrapper allows the same layer to be applied to each time step of the input sequence independently. This is essential for processing sequences of video frames

Convolutional Layers

4 Convolutional Blocks

- **Feature Extraction**: The Conv2D layer extracts local features from the input data.
- **Dimensionality Reduction**: The MaxPooling2D layer reduces the spatial dimensions, making the subsequent layers more efficient.
- **Regularization**: The Dropout layer helps prevent overfitting, ensuring that the model generalizes well to new data.

Model explanation

Model explanation

Flatten Layer

Flattens the output of the convolutional layers to a 1D vector for each time step.

LSTM Layer

Adds an LSTM layer with 32 units to capture temporal dependencies in the sequence of flattened vectors.

Dense Layer

Adds a dense layer with a number of units equal to the number of classes and a softmax activation function for classification.

Model Summary

Layer (type)	Output Shape
time_distributed_3 (TimeDistributed)	(None, 20, 64, 64, 16)
time_distributed_4 (TimeDistributed)	(None, 20, 16, 16, 16)
time_distributed_5 (TimeDistributed)	(None, 20, 16, 16, 16)
time_distributed_6 (TimeDistributed)	(None, 20, 16, 16, 32)
time_distributed_7 (TimeDistributed)	(None, 20, 4, 4, 32)
time_distributed_8 (TimeDistributed)	(None, 20, 4, 4, 32)
time_distributed_9 (TimeDistributed)	(None, 20, 4, 4, 64)
time_distributed_10 (TimeDistributed)	(None, 20, 2, 2, 64)
time_distributed_11 (TimeDistributed)	(None, 20, 2, 2, 64)
time_distributed_12 (TimeDistributed)	(None, 20, 2, 2, 64)
time_distributed_13 (TimeDistributed)	(None, 20, 1, 1, 64)
time_distributed_14 (TimeDistributed)	(None, 20, 64)
lstm (LSTM)	(None, 32)
dense_1 (Dense)	(None, 4)

Figure: Layers and output shape of the model

Read Video File

Use cv2.VideoCapture to read the video file.

Determine the interval at which frames will be extracted

- Retrieve the total number of frames in the video.
- Determine the interval at which frames will be extracted. This is calculated as the total number of frames divided by the desired sequence length.

Process each frame in the video by iterating through them to generate the output frame list

- Set Frame Position: Set the current frame position (FrameCounter * SkipFrameWindow).
- Read Frame: Read the frame at the current position.
- Resize Frame: Resize the frame to the desired dimensions (ImageHeight, ImageWidth) using cv2.resize.
- Normalize Frame: Normalize the frame by scaling pixel values to the range [0, 1] using ResizedFrame / 255.
- Append Frame to List: Append the normalized frame to FrameList.

Extracts frames from a video

Training Process

Model Compilation

- Loss Function: categorical_crossentropy for multi-class classification.
- Optimizer: Adam for efficient training.
- Metrics: accuracy to evaluate performance.

Model Training

- Data: Uses UCF50 dataset.
- Epochs: Trains for up to 70 epochs.
- Batch Size: Uses a batch size of 4.
- Shuffle: Shuffles the training data before each epoch.
- Validation Split: Uses 20% of the training data for validation.

Demo

Conclusion

Model Potential

- The model shows potential in predicting single actions in short videos containing only one action.
- Demonstrates the ability to process video data efficiently.
- Achieves high accuracy in classifying actions.

Challenges

- Processing time needs improvement.
- Complexity increases with videos containing multiple actions or longer durations.
- Further research and model upgrades are required to address these challenges.

Future Directions

- Improve model performance.
- Optimize training time.
- Enhance the ability to handle larger datasets.

References

https://github.com/BaranidharanB/Human-Pose-Recognition-using-LSTM-LRCN-CNN

https://arxiv.org/abs/1411.4389

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Grateful for your Attention!

Q&A