# Deep Learning Project – Gesture Recognition

# Submitted by Shubhabrata Paul Choudhury (DS C68)

## **Problem Statement:**

This project involves building a 3D Convolutional Neural Network (CNN) to correctly recognize hand gestures by a user to control a smart TV.

The objective of this projects is to build a hand gesture recognition model that can be hosted on a camera installed in a smart TV that can understand 5 gestures.

The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

👍 **Thumbs up** → Increase volume

👎 **Thumbs down** → Decrease volume

👈 **Left swipe** → Rewind 10 seconds

👉 **Right swipe** → Fast-forward 10 seconds

✋ **Stop** → Pause the movie

**Dataset Overview:**

The dataset consists of several hundred short video clips, each categorized into one of the five gestures. Each video is about 2-3 seconds long and broken into 30 sequential frames. These videos were recorded using different webcams, leading to two resolutions: **360x360** and **120x160** pixels. The dataset simulates real-world usage conditions similar to a smart TV's camera setup.  
  
Data Source : [Google Drive Link](https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL)

**Neural Network Architectures Used:**

Two primary deep learning architectures were considered for video-based gesture recognition:

1. **CNN+ RNN (LSTM/GRU)**

This hybrid approach processes each frame using a **2D CNN** to extract feature vectors, which are then passed to an **RNN (LSTM or GRU)** to capture temporal dependencies.

* **Why LSTM/GRU?** These models effectively understand sequential information. GRUs are computationally more efficient than LSTMs since they have **three** gates instead of four, leading to **faster training** without significant accuracy loss.
* **Transfer Learning**: Pre-trained CNNs like **ResNet** was used to extract image features, reducing training time while leveraging existing powerful models.

**2. 3D Convolutional Networks**

Instead of treating videos as separate image frames, this model processes them as a **4D tensor** (Height × Width × Time × Channels).

* **3D Convolutions** extend traditional 2D convolutions by applying filters across the **time axis**, allowing the network to capture motion features.
* While powerful transfer learning, **3D CNNs require higher computational resources** compared to CNN-RNN architectures.

**Data Ingestion Pipeline & Custom Generator**

To efficiently handle video data, a **custom batch data generator** was implemented using Python's **generator functions**.

* Unlike Keras' built-in image generators, this custom generator efficiently loads **video sequences** in memory-friendly batches, leveraging **lazy evaluation**.
* Generators optimize **memory usage, execution speed, and batch-wise gradient descent**, which is crucial for large datasets.
* The custom generator supports various data formats (e.g., images, CSV files, audio), making it **versatile and scalable**.

**Experiments & Model Selection**

Experiments were conducted to find the most effective model:

* **Training Challenges**:
  + Initial models were all overfitting due to lesser input files
  + Sometimes, validation loss stagnates or increases despite multiple epochs, indicating a plateau in learning.
  + Reducing the learning rate (e.g., **0.0002 for Adam optimizer**) helped optimize performance.

**Summarizing the approach**

| **Model Architecture** | **Key Layers** | **Validation Accuracy** | **Decision & Explanation** |
| --- | --- | --- | --- |
| **CNN + LSTM** | Conv2D + LSTM + Dense | **87%** | Selected as the final model due to optimal balance of performance and efficiency (<1M parameters). |
| **CNN + GRU** | Conv2D + GRU + Dense | 79% | Performed well but slightly lower accuracy than LSTM. Comparable in efficiency. |
| **3D CNN** | Conv3D + MaxPooling3D + Dense | 83% (Best of augmented) | Higher complexity but underperformed compared to CNN-LSTM. Performance improved with augmented data. |
| **CNN + Transfer Learning (ResNet)** | Pre-trained ResNet50 + Dense | 75% | Too many parameters (~25M), making training sluggish with suboptimal accuracy. |

* **Final Model Chosen**:
  + **CNN + LSTM model** was selected as the best performer with **<1M parameters** and **87% validation accuracy**.
  + **Transfer learning with ResNet** was tested but was computationally heavy (**25M+ parameters**) and slower, yielding only **75% accuracy**.
  + The CNN-LSTM model provided an optimal balance between **accuracy and computational efficiency**.

**Scope for Improvement**

Potential enhancements for future work:

* Fine-tuning learning rate & regularization for better generalization.
* Dropout in LSTM layers to reduce overfitting (not fully explored).
* Bidirectional LSTMs for improved sequence modeling
* Vision Transformers (ViTs) for advanced video representation learning.

The focus would be on better regularization, dropout strategies, and temporal sequence learning to enhance performance.