Swift-Spatio Flow: A Human Action Recognition Model Using 3D CNN-LSTM

A project report submitted in partial
fulfillment of the requirements for the
degree of

Bachelor of Engineering
in
Computer Engineering

by

Ian Joseph K(8948)
Aryan Patil (8962)
Abhishek Raje (8965)
Ramani Harsh Anilkumar (8966)

Under the guidance of

Prof. Merly Thomas

DEPARTMENT OF COMPUTER ENGINEERING

Fr. Conceicao Rodrigues College of Engineering, Bandra (W), Mumbai - 400050

University of
Mumbai
2022-23

This work is dedicated to my family.
I am very thankful for their motivation and support.

Internal Approval Sheet

CERTIFICATE

This is to certify that the project entitled **"HUMAN ACTION RECOGNITION"** is a bonafide work of **Ian Joseph K (8948), Aryan Patil (8962), Abhishek Raje (8965), Ramani Harsh Anilkumar (8966)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor in Computer Engineering**.

(Name and sign)
Supervisor/Guide

(Name and sign)
Head of Department

(Name and sign)
Principal

Approval Sheet

Project Report Approval

This project report entitled **Human Action Recognition** by **Ian Joseph K, Aryan Patil, Abhishek Raje, Ramani Harsh Anilkumar** is approved for the degree of Bachelor of Engineering in Computer Engineering.

Examiner1.
_____

Examiner
2._____

Date:
Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Ian Joseph K (8948) **(sign)**

Aryan Patil (8962) **(sign)**

Abhishek Raje (8965) **(sign)**

Ramani Harsh Anilkumar (8966) **(sign)**

Date:

Place:

Abstract

Human Action Recognition(HAR) remains to be one of the most sought after research areas in the field of Computer Vision because of its potential in assisting a vast array of tasks like CCTV surveillance, tools that aid the blind, self-driving cars, and data analysis in sports to name a few. Despite the promising scope this setup has, there are quite a few challenges the existing systems face. Unclear CCTV footage makes it tough for the security personnel to extract intel from the videos or draw unambiguous insights and assisting the blind is one the largest stumbling blocks even today. Yet these problems are not solved using the existing models as a preference because of the complexity of the architecture, poor performance in terms of being computationally expensive, and inability to reach a convincing accuracy. Therefore we are here proposing a novel model in this paper as a solution to the above drawbacks. We treat it as a video classification problem and thus the use of deep neural networks allows the extraction of spatial features from the individual frames from the video. The Long Short Term Memory network is needed so that this can be done to a sequence of frames or extract temporal features from the video. Finally, the average of predictions is taken as input by the Softmax layer to give the final output which increases the accuracy of classifying equivocal images or frames in the video. The neural network generally used is Convolutional Neural Network 2D(CNN 2D) but when compared to the 3D(Three dimensional) neural network, it's unable to process temporal features. Additionally, 3D CNN allows to extract deep and discriminative features from frames which the 2D CNN's are more likely to miss out on. Yet, 3D CNNs are not popularized much in HAR primarily because of the slow training time and require large amounts of data to train on. Thus to use them is computationally expensive and time is wasted in either collecting large amounts of data as well as labeling it. Our model, Swift-Spatio Flow is proposed with the intention to overcome these snags allowing a combination of 3D CNN and a variant of LSTM, with optimal parameters and robust architecture to reduce computational cost by a significant margin, train the model on smaller data, and generate accuracy better than the other existing 2D and 3D models. We have used the UCF 50 dataset that contains 50 action classes and roughly 6600 action videos to train four very popular models in Human Action Recognition to compare the accuracy, performance, cost, and complexity of the models with our proposed model trained on the same set of data. Accordingly, after drawing the most noteworthy conclusions, we have lodged a few hardware integrated projects that are based on the novel model and have capability of being a pragmatic solution to multiple real world challenges.

Acknowledgments

We have great pleasure in presenting the report on **"Human Action Recognition"**. We are extremely obliged to get an opportunity to work on a project that has taught us so much from a technical and non-technical perspective. To thank for that, we have our guide and mentor Prof. Merly Thomas, C.R.C.E, Bandra (W), Mumbai, under whose guidance and with whose suggestions regarding the line of this work, we have been able to carry out this project.

We thank Dr. Sujata Deshmukh, Head of Computer Engineering Dept., Principal and the management of C.R.C.E., Mumbai for encouragement and providing necessary infrastructure for pursuing the project.

We are also thankful to all our other teachers and all the non teaching staff at Fr. Conceicao Rodrigues College of Engineering, Bandra for their support has helped us to complete our project.

Ian Joseph K (8948)
Aryan Patil (8962)
Abhishek Raje (8965)
Ramani Harsh Anilkumar (8966)

Date

Contents

List of Figures

List of Tables

Glossary

HAR Human Action Recognition. 35

CNN Convolutional Neural Network, which is a type of artificial neural network that is used in computer vision tasks such as image and video recognition. 66

LSTM Long short-term memory is an artificial neural network used to learn, process, and classify sequential data. 60

ConvLSTM2D Convoluted LSTM(2D) model is an enhanced version of the original LSTM models. 9

LRCN Long-term recurrent convolutional network is a model that is both spatially and temporally deep. 1

Chapter 1

Introduction

1.1 Overview of Human Action Recognition:

Human Action Recognition (HAR) can be referred to as the activity of identifying and stating various acts carried out by people by subjecting Artificial Intelligence (AI) systems to a plethora of data accumulated in datasets that's gathered by utilizing different centralized/decentralized sources[1]. This is done due to the improvement found in lifestyle by using novel technology. The proponent why HAR's evolving is because of the quick burgeoning of approaches like Machine Learning, Computer Vision, Natural Language Processing and Human Computer Interaction, the most promising sub-fields of AI. This makes it possible to employ such devices across a range of application domains. The previous variants of the models were based on only one sole image or a brief sequence of images[2], but the progress in the field of AI has handed us increasing chances. The constant improvement in accuracy of AI based models is devoted to exponential progressions in reducing computational costs in terms of time and space. The development of HAR is positively correlated with the development of AI which increases the scope of HAR across a range of application-based domains. The HAR domain has been able to better extract meaningful features from the raw sensor data due to the establishment of deep learning(DL) to this field[3]. This made the HAR- based tasks to be designed as plug- and- play based. Although this task is moving towards becoming more black-box oriented, much deeper comprehension is still needed for ensuring that the three-legged stool is sturdy and functional[4].

Figure 1.1: Illustrating various human activities

When trying to fete mortal conditioning, one must ascertain a person's behavior. A person's kinematic state must be known when trying to celebrate mortal conditioning in order for the device to effectively recognise this effort[5]. Mortal conditioning like “ jumping", `` walking" ” and “ handling, ” are fairly easy to fete as they arise naturally in our diurnal lives. Contrasted with, more difficult conditioning, similar to “ shelling an apple, ” are further tougher to recognise[6]. Difficult conditioning can be further divided into other straight-forward conditioning, that are usually simpler to feat. This method of breaking down a complex problem to easier sub-problems is often used in solving many problems in multiple domains of computer programming[7].

1.2 Motivation:

The prospect of multiple applications—both offline and online—has sparked our interest in this subject. Activity recognition systems have been created, thanks to investigations in peripheral assisted living. The aged population's standard of living as well as their medical services are improved by such systems[8]. By evaluating the information, the project's primary goal is to identify human behaviors including walking and running.

This project of human action recognition spans multiple disciplines. It has drawn considerable interest from two important scientific work areas. Artificial intelligence and natural language processing[9].

It necessitates the fusion of two disciplines, namely computer vision and natural language processing[10].

- **Natural Language processing:**

  We need NLP to understand and analyze the question and to generate the answer[11].

- **Computer Vision**

  In order to answer the question from the image, the system must be able to detect various features of objects present in the image[12].

  So, a good HAR. system must be capable of solving a broad spectrum of typical NLP and CV tasks[9].

1.3 Objectives:

- Studying different data sets and selecting an appropriate one for the project.
- Performing preprocessing on the data set using feature recognition.
- Implementing a novel model for HAR.
- Evaluation and comparison of HAR. models

1.4 Scope:

- Research is ongoing in the field of cellular device technology, which now includes more sensors to support cutting-edge applications[5].
- Driverless cars can sense and predict foot activity far more comprehensively when human activity is recognized from video[6].
- Video surveillance is now extremely important for ensuring civic safety. Governments frequently install CCTV cams to keep an eye on crowd behavior in order to ensure civic peace[7].
- VQA (Visual Question Answering), the HAR's replacement, can be used to communicate with machines in human language[4].
- The "AI-complete" task of human action recognition (HAR.) necessitates knowledge and experience with multiple tasks outside of a specific field[1,3].
- One of the components of a smart system developed to address the issue of interactive question-answering issues may incorporate a human action recognition system. Ex: To enable communication for blind users via images[4].
- Everyday life routines frequently involve a variety of low-cost, discreet sensors like accelerometers or microphones[5].

Chapter 2

Literature Review

2.1 Existing Models

2.1.1 HAR. implementation using the FR-DL Method and testing on videos[13]:

Summary: The proposed FR- DL system assisted us in recognising complex conduct using spatio-temporal information, which were hidden in successional patterns and features. The inspection and evaluation was carried out using metrics like the measure of delicacy, time complexity, and confusion matrix, and the overall results demonstrated that the alleged FR-DL system surpassed its six previous exceptional field explorations in terms of recognising mortal action. The model consists of two dimensional neural networks by laying convolutional neural network layers and finally the LSTM layer for classification. But the model has poor architecture and is less accurate.

Drawbacks: The proposed method involves a complex hybrid approach that combines both hand-crafted features and deep learning techniques. This approach may be less interpretable than purely deep learning methods, as it involves multiple layers of processing and feature extraction.

The proposed method was evaluated on a relatively small dataset of only few action classes, which may limit the generalizability of the results to other action recognition tasks or datasets.

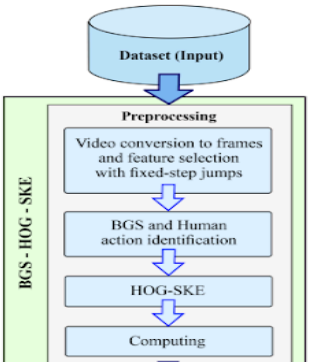

Figure 2.1: HAR system with FR-DL

2.1.2 Distances evolution analysis as a methods for understanding spatial features[14]:

Summary: In this essay, we looked at a useful human-object interaction strategy. Results of interactions between humans and entities that can adapt to changes in position and speed have been reported. This approach is suitable for online recognition due to a number of factors . In the first stage, we take care of the various stances. We have suggested the object-joint dimensions as well as the inter-joint dimensions as lower categories. The length tracks are then subjected to a Riemannian analysis in order to account for rate differences.The proposed framework uses a two-stage approach, where the first stage detects human-object pairs and the second stage classifies the interactions between them. The authors argue that existing methods for human-object interaction detection do not consider the impact of object distance, which can lead to inaccurate detection results.

Drawbacks: The proposed method relies heavily on the availability and accuracy of human pose and object detection, which can be challenging in some scenarios, such as low-resolution or occluded images/videos, or complex scenes with multiple objects and people.

The method is designed for human-object interaction detection, which is a specific subtask of action recognition that focuses on identifying interactions between people and objects. This

may limit the applicability of the method to other action recognition tasks that do not involve such interactions.

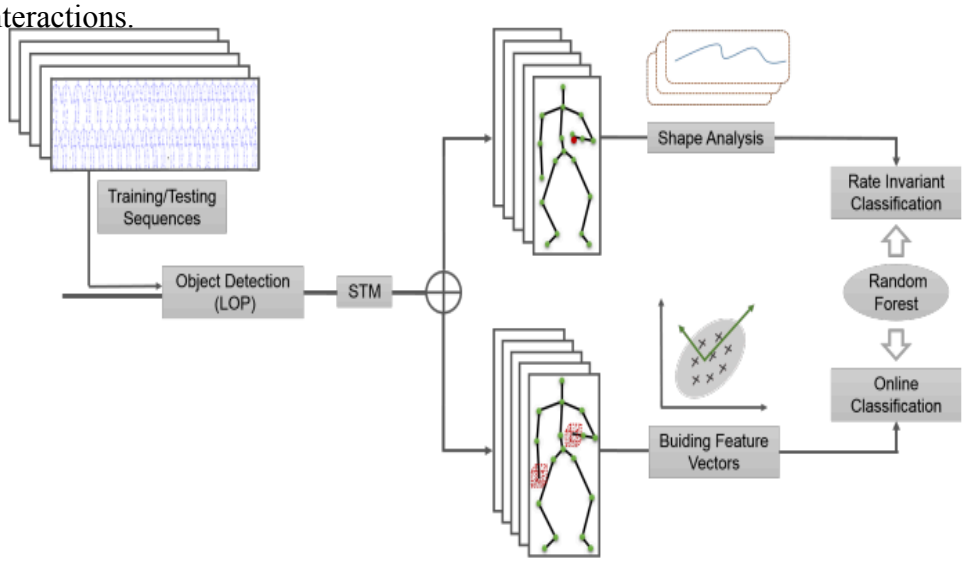

Figure 2.2: Four Steps showing of CNN and SVM model

2.1.3 Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet[15]:

Summary: Here for this paper, they used recent video datasets to analyze the designs of several neural networks with space-time three dimensional kernels. The outcomes of the experiments concluded that:

(a) When using ResNet-18 for training, we observed noticeable overfitting in UCF-101, ActivityNet, and HMDB-51 datasets, but it worked well for the Kinetics dataset.

(b) Interestingly, the Kinetics pre-trained model outperformed complex 2D architectural designs on HMDB-51 and UCF-101 datasets. For instance, while the ResNeXt-101 model achieved 94% accuracy on UCF-101, it only reached 70% on HMDB-51.

(c) Our findings suggest that the Kinetics dataset is suitable for training deep 3D neural networks, such as ResNets with over 150 layers.

Drawbacks: The proposed method requires a large amount of training data and computational resources, which may not be feasible for some applications or on resource-constrained devices. The use of 3D convolutions can lead to higher model

complexity and longer training times compared to 2D convolutions, which can be a practical issue for some applications. The authors did not perform an extensive analysis of the impact of hyperparameters, such as learning rate, batch size, or network architecture, on the performance of their method.

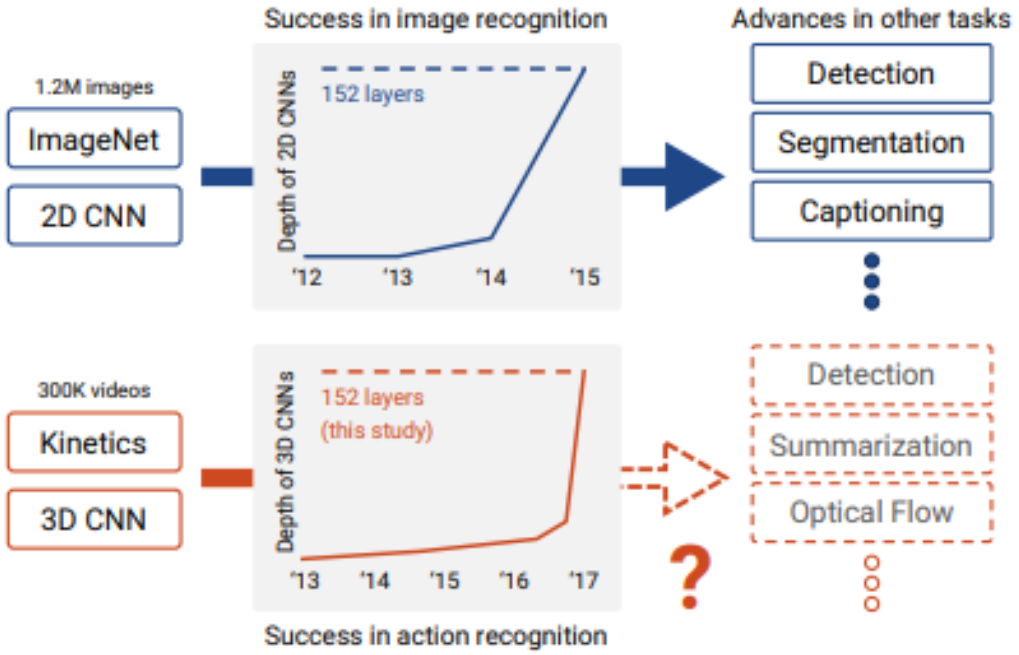

Figure 2.3: Analysis of 3D CNN on Kinetics Dataset

2.1.4 Bridge Deep learning with mobile phones[16]:

Summary: In this paper, the authors propose a system for Human Activity Recognition (HAR) using smartphones and deep learning. The system uses smartphone sensors, such as accelerometer and gyroscope, to collect data on human activities, and employs CNNs LSTM networks for activity recognition.The authors also evaluate the impact of different factors, such as the number of sensors used, the window size, and the input features, on the performance of their system. They show that using multiple sensors and larger window sizes can improve the accuracy of the system, and that using raw data instead of preprocessed features can lead to better performance.The authors collect and preprocess data from the publicly available UCI HAR dataset, which consists of six activities performed by 30 subjects, and use it to train and evaluate their system.Overall, the paper demonstrates the effectiveness of deep learning models for HAR using smartphone sensors and highlights the

importance of optimizing the system's parameters and input features for achieving high accuracy.

Drawbacks: The proposed model architecture consists of only two convolutional layers and one fully connected layer, which may limit the capacity of the model to capture complex patterns in the sensor data.The authors did not provide a clear explanation of why they chose the specific architecture for their model, or how they optimized the hyperparameters.The paper did not conduct a comparative analysis of the proposed architecture with other existing architectures, which could limit the understanding of its performance relative to other approaches.

2.1.5 Why 3D CNN has an upper hand dealing with spatiotemporal data[17]:

Summary: In this paper, the authors propose 3D convolutional networks (3D CNNs) for learning spatiotemporal features from video data. 3D CNNs extend traditional 2D CNNs by adding an additional temporal dimension to capture the temporal evolution of features over time. The authors demonstrate the effectiveness of 3D CNNs on various video datasets, including UCF101, HMDB51, and Sports1M, and show that they outperform previous state-of-the-art methods. The authors also analyze the impact of different network architectures and training strategies on the performance of 3D CNNs. They conclude that 3D CNNs are a powerful tool for learning spatiotemporal features from video data and have potential applications in video recognition, action recognition, and video analysis.

Drawbacks: The model requires a large amount of computational resources and time to train due to the use of 3D convolutions, which can limit its practicality in certain applications. The paper did not evaluate the interpretability or explainability of the model, which can be important for some applications where transparency is required. The authors did not investigate the robustness of the model to common problems such as occlusions, noise, or variations in viewpoint, which can impact the model's performance in real-world scenarios.

2.2 Datasets for Human Action Recognition:

a) UCF-101[18]:

The data set known as UFC-101 is an expansion of the previously existing UCF50 data set, which contained 50 categories of actions. This newer data set is particularly challenging, as it includes 101 different categories of actions and a wide range of variations within the videos themselves, including camera motion, varied object appearance, differing poses, object scale, varying viewpoints, complex backgrounds, and differing illumination conditions. In total, this data set includes over 13,000 videos.

b) HMDB-51[19]:

The dataset contains a total of 6,849 videos and includes 51 distinct categories of actions, such as "jump", "laugh", and "walk". Every action category in the dataset contains at least 101 video clips. To evaluate the dataset, three distinct sets are used for training and testing, with each set containing 70 video clips for every action category and 30 video clips for testing purposes. The final performance of the dataset is measured based on the average accuracy across all three sets.

c) ActivityNet[20]:

The ActivityNet dataset comprises 849 hours of YouTube videos and includes a wide range of 200 different types of activities. This dataset poses a significant challenge, as it is currently the largest dataset in terms of the total number of action classes and the number of videos included in the collection.

d) Kinetics[21]:

This dataset is a sizable collection for the purpose of Human Activity Recognition, which includes roughly 500,000 high-quality videos that encompass a broad range of 600 action categories. Each video in the dataset is around 10 seconds in duration. Lei Wang et al, present a Comparative Analysis of Recent Action Recognition Approaches using Kinetics database[18]

e) RGB-D dataset[22]:

The EPFL RGB-D Pedestrian dataset consists of over 5000 RGB + Depth images acquired from an RGB camera and Kinect V2 sensor setup.

f) LEMMA[23]:

A Multi-view Dataset for **LE**arning **M**ulti-agent **M**ulti-task **A**ctivities. With the use of ground-truth labels for compositional atomic actions and the tasks these are connected with, this dataset intends to investigate the essence of complex human behaviors in a goal-directed, multi-agent, multi-task environment.

g) MSRDaily Activity 3D[24]:

The dataset comprises recordings of daily activities captured by Kinect devices, encompassing a wide range of actions, including drinking, eating, reading a book, using a cell phone, writing on paper, operating a laptop, using a vacuum cleaner, cheering, sitting still, tossing paper, playing games, lying on a sofa, walking, playing guitar, standing up, and sitting down.

h) UCF Sports dataset[25]:

The UCF Sports dataset comprises a collection of actions observed in a variety of sports commonly broadcast on television channels such as the BBC and ESPN. The dataset features 150 sequences, each with a resolution of 720 x 480, and includes a diverse range of actions depicted from different scenes and viewpoints

Chapter 3

Problem Statement

Human activity recognition is the problem of classifying sequences of data collected from for a few specific movements to be identified. This project is about identifying and predicting the activity someone is performing based on recorded data.

3.1 Challenges in HAR:

1. Feature Extraction: Extracting meaningful features from sensor data is a significant challenge in HAR. It is crucial to extract the relevant information while eliminating irrelevant noise.
2. Model Selection: Choosing the right machine learning algorithm and tuning its parameters is critical to achieving high accuracy in HAR.
3. Real-time Recognition: Real-time recognition of human activities is challenging as the system must process the data quickly and accurately to provide timely feedback or make decisions.
4. Data Collection: Collecting reliable and diverse data from multiple sensors is a significant challenge in HAR. The quality and quantity of the data collected can significantly impact the accuracy of recognition.
5. Class Imbalance: A class imbalance problem exists in HAR where some activities occur less frequently than others, making it harder to recognize them accurately.

3.2 Solutions to these challenges:

1. Real-time optimization: To overcome the challenge of real-time performance, it is important to optimize models for speed and efficiency. This can involve using lightweight models, reducing the number of computations required, and optimizing the model for the target hardware.

2. Feature extraction: Another approach in HAR is to extract features from the video frames, such as optical flow, histograms of oriented gradients (HOG), or deep features from pre-trained convolutional neural networks (CNNs). These features can then be used to train a machine learning model, such as a support vector machine (SVM) or random forest classifier.
3. Long short-term memory (LSTM): LSTMs are a type of recurrent neural network (RNN) that can learn to model the temporal dependencies between video frames and are particularly effective for recognizing actions that occur over longer time periods.
4. 3D convolutional neural networks (CNNs): 3D CNNs can learn spatio-temporal features from video data and have been shown to be effective for HAR. They can achieve state-of-the-art performance on benchmark datasets.
5. Ensembling: Ensembling involves combining the predictions of multiple models to improve accuracy. This can involve training multiple models with different architectures, hyperparameters, or training data, and then combining their predictions using techniques such as majority voting or stacking.
6. Semi-supervised learning: Semi-supervised learning can be used to overcome the challenge of limited labeled data by training a model on a combination of labeled and unlabeled data. This can improve its generalization performance and reduce the amount of labeled data required.

It's important to note that the specific solutions used in HAR will depend on the specific challenges of the application and the resources available.

Chapter 4

Project Description

4.1 Overview of the project:

The goal of our deep learning project was to develop a model for human action recognition using the UCF50 dataset. We started by creating our own pre-processing code that cleaned, resized, and normalized the videos in the dataset, preparing them for training and testing. Next, we trained four models, including three existing models and a novel 3D CNN + LSTM model that we created. During training, the models learned to extract relevant features from the videos, and we evaluated their performance using various metrics, such as accuracy, precision, recall, and F1-score. By comparing the performance of the models, we were able to determine which approach was most effective for human action recognition.

Overall, our deep learning project contributed to the field of computer vision and deep learning by developing a model that could accurately recognize human actions. Our use of a novel 3D CNN + LSTM model showcased our ability to innovate and experiment with different approaches to achieve our goal. The evaluation metrics we used helped us to measure the performance of the models and provided insights into the strengths and weaknesses of each approach.

4.2 Module and Library Description:

The modules and libraries used in this project are given below.

4.2.1 Keras Tensorflow library:

Keras and TensorFlow are two powerful open-source libraries that can be used together to create complex machine learning models[25]. Keras provides a user-friendly and high-level interface to build and train deep learning models[26], while TensorFlow provides the underlying computational engine to perform complex mathematical operations on

tensors.Using Keras with TensorFlow allows developers to take advantage of the best of both libraries: Keras simplicity and ease-of-use, and TensorFlow's powerful computational abilities[27].

This combination enables the creation of highly customizable and efficient deep learning models with ease.Keras provides a variety of pre-built layers and models that can be easily combined to create new models, and it also allows users to define their own custom layers and models[26]. TensorFlow, on the other hand, provides a low-level interface to perform operations on tensors, allowing for a high degree of customization and control[27].

Overall, Keras and TensorFlow provide a comprehensive set of tools for machine learning and deep learning, making it a popular choice among developers and researchers alike.

4.2.2 Modules used:

1. os: This module allows Python to interact with the file system and provides methods for working with directories, files, and paths.
2. cv2 (OpenCV): This is a Python library used for image and video processing that provides various functions for manipulating, analyzing, and understanding visual data.
3. pafy: This is a Python library that allows easy access to YouTube content and metadata, such as video URLs and duration.
4. math: This module provides various mathematical functions and constants for performing mathematical operations in Python.
5. random: This module provides functions for generating random numbers and sequences, shuffling sequences randomly, and making random selections.
6. numpy: This library is used for working with arrays and matrices, and provides a large collection of mathematical functions to operate on these structures.
7. datetime: This module provides classes for working with dates and times, including formatting and parsing, time intervals, and timezone support.
8. tensorflow: This is an open-source machine learning library used for building and training deep neural networks.
9. deque (from collections): This module provides a double-ended queue data structure, which is useful for implementing queues and stacks.
10. matplotlib.pyplot: This is a Python plotting library used for creating static, animated, and interactive visualizations.

11. sklearn.model_selection: This module provides tools for splitting datasets into training and testing sets, and for evaluating the performance of machine learning models.
12. tensorflow.keras.layers: This module provides various types of layers for building deep neural networks, such as dense, convolutional, and recurrent layers.
13. tensorflow.keras.models: This module provides a way to define neural network models as a sequence or graph of layers.
14. tensorflow.keras.utils: This module provides utilities for working with Keras models, such as transforming data to fit a model's input shape, one-hot encoding of categorical variables, and early stopping during training.
15. tensorflow.keras.callbacks: This module provides callbacks that can be called during training to perform certain actions, such as saving the model after each epoch or stopping training early if the validation loss stops improving.

4.3 Diagrams:

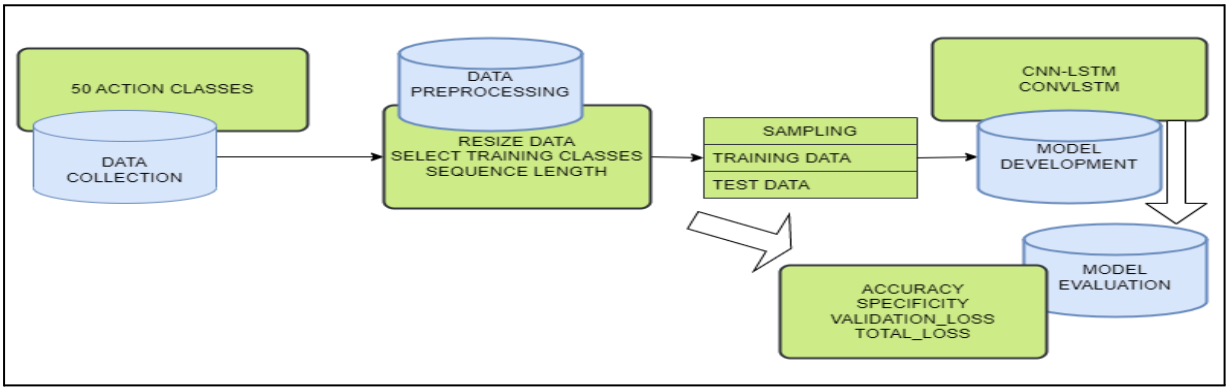

Fig 4.1 Steps involved in data preprocessing

The second flow diagram outlines the data preprocessing steps for the ML model. The diagram shows the following steps: importing libraries and dataset, data visualization, initializing constants, frame extraction, creating a dataset, and splitting the dataset into training and testing sets. These steps are all necessary to prepare the data for the ML model to be trained on.

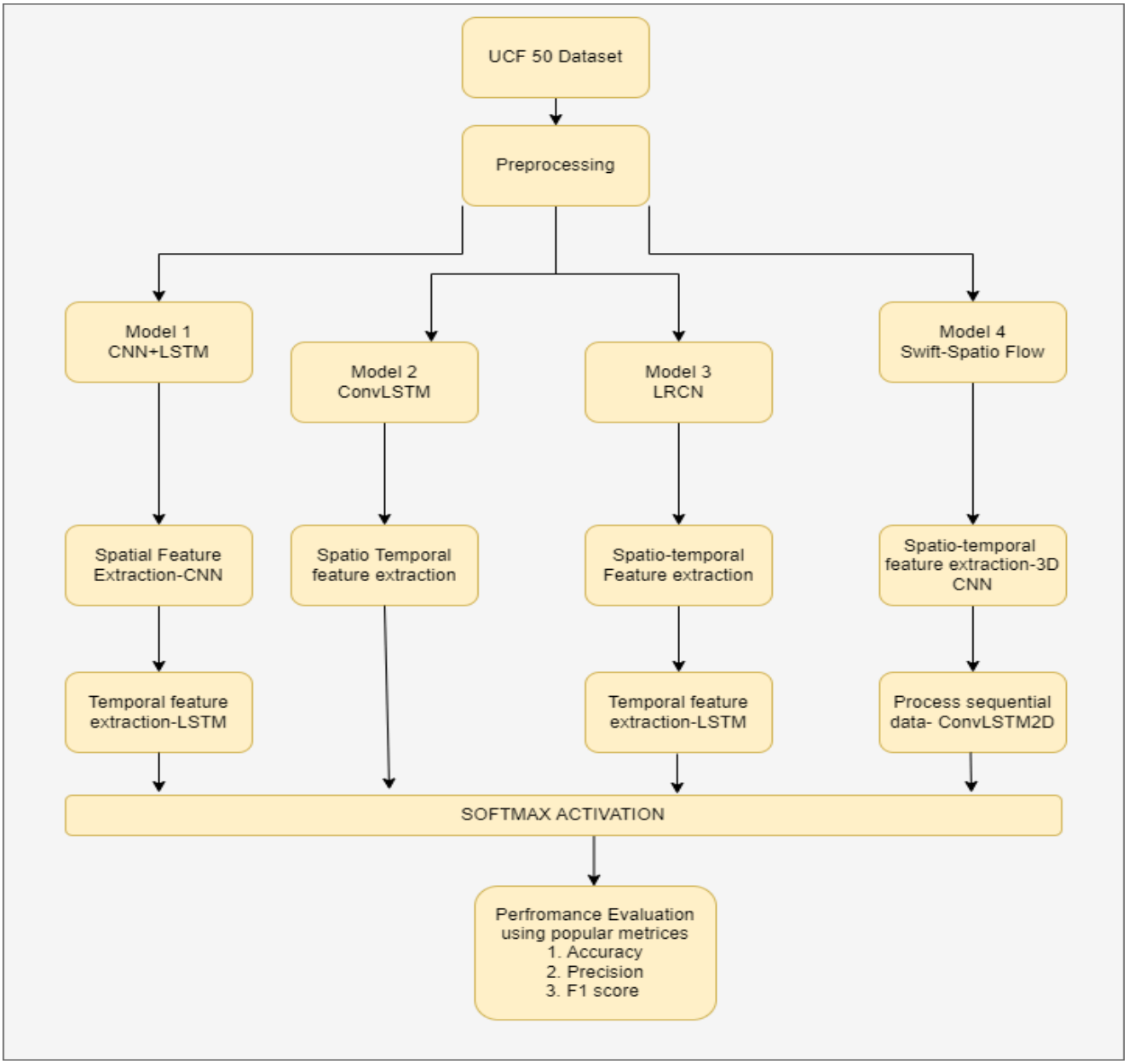

Fig 4.2 Process Flow Diagram

The diagram illustrates the steps involved in assessing the CNN model's effectiveness. Firstly, the pre-trained model and the test dataset are loaded. The model's predictions for the test dataset are then obtained, and these predictions are compared against the true labels to create a confusion matrix. Finally, the model's performance is evaluated using various metrics, including accuracy, precision, recall, and F1 score, which are calculated from the confusion matrix. The flow diagram indicates that the evaluation results can be used to enhance the model by altering its parameters or architecture.

conv1_input	input:	[(None, 20, 64, 64, 3)]
InputLayer	output:	[(None, 20, 64, 64, 3)]

Figure 4.3: Layered Architecture of Swift-Spatio Flow

The model architecture diagram provided in the document shows a Convolutional Neural Network (CNN) architecture with 3 convolutional layers, each followed by a max-pooling layer and a dropout layer. The output of the third convolutional layer is flattened and passed through a dense layer followed by a dropout layer, and finally, the output layer with the number of units equal to the number of classes in the dataset.

4.4 Pseudo Code:

```
# Importing Required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout

# Load the data from the given CSV file
data = pd.read_csv('filename.csv')

# Data preprocessing
# Check for missing data
if data.isnull().sum().any():
    # Remove rows with missing data or fill missing data using imputation
    data = data.dropna() # or use data.fillna(value) to fill missing data

# Check for outliers
if data.select_dtypes(include=[np.number]).apply(lambda x: np.abs(x - x.mean()) / x.std() > 3).any().any():
    # Remove outliers using appropriate method like Z-score, IQR or any other method
```

```
# Convert data to numpy arrays
X = data.drop('target', axis=1).values
y = data['target'].values

# Normalize the data
X = X / 255.0

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Reshape the data for CNN
X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)

# Create and train a CNN model
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_train, batch_size=128, epochs=10, verbose=1,
validation_data=(X_test, y_test))

# Evaluate the model
score = model.evaluate(X_test, y_test, verbose=0)
print("Test Loss: ", score[0])
print("Test Accuracy: ", score[1])
```

```
# Plot the accuracy and loss curves
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Training', 'Validation'], loc='lower right')
plt.show()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Training', 'Validation'], loc='upper right')
plt.show(
```

Chapter 5

Implementation Details

5.1 Methodology

For training all the models, we use the ***UCF 50 dataset***. This dataset consists of 50 action classes where each class has multiple videos of the category. This is a very popular and sufficient dataset when dealing with two dimensionals Convolutional Networks. Although it seems to be insufficient for three dimensional convolutions, we aim at proposing a unique model that trains well on this dataset.

STEPS:

1. Preprocessing: Preprocess the dataset and divide it into 75:25 ratio.
2. Define the model architecture: Two existing models, one hybrid/variant of an existing model and one novel model are implemented.
    - CNN+LSTM (Existing)
    - ConvLSTM2D (Existing)
    - LRCN Hybrid (Existing)
    - Swift-Spatio Flow (CNN 3D + LSTM) (Novel)
3. Training: Train all the models under similar constraints and parameters.
4. Testing: Test the trained model on random youtube videos and compare the results.

5.1.1 Preprocessing:

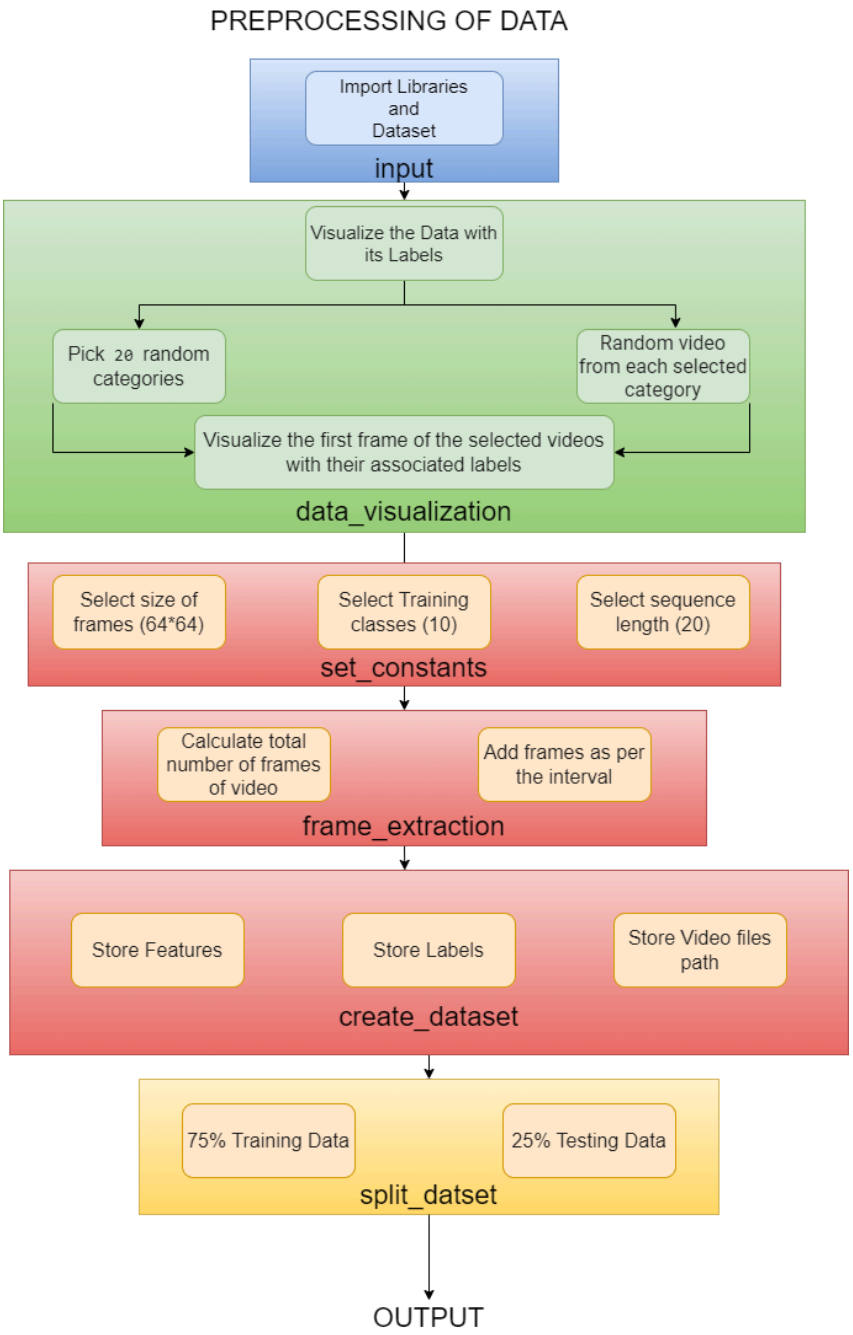

Figure 5.1: Illustrating various steps in Preprocessing

5.1.1.1 Import libraries and dataset:

We begin by importing libraries like- tensorflow keras which we will be using to define the architecture of the proposed model. We need the robust scikit-learn toolkit to split the training and testing data. The UCF 50 dataset used to train the model contains 50 action categories, each of which contains 25 groups of videos.

5.1.1.2 Data Visualisation:

Data Visualisation gives us the overview of the dataset-UCF50.25 arbitrary values are now created and stored in a list. As there are Fifty categories in the dataset, the values will range from 0 to 50. Select a video file at random from the collection of all the video assets that are present in the previously chosen Class Directory by iterating over the values that were created . To better comprehend the dataset, the first frame of the video clip is now read and shown with a label. It is therefore an important step when we further split data into training and testing.

5.1.1.3 Initialize constants, frame extraction and create a dataset

In order to resize the frames to make the training uniform, pass 4 parameters for input_shape. These are *IMAGE_HEIGHT, IMAGE_WIDTH, IMAGE_DEPTH and SEQUENCE_LENGTH.* Set these values to 64,64,64,20 initially and can be increased for better accuracy , although at the cost of being more computationally expensive.

We create a function for frame extraction. It will generate an array with the video's scaled(64*64) and normalized frames using the location supplied as a parameter(args). Depending on the length of the sequence, it reads the video, one frame at a time, and adds them to an array.

The create dataset function calls the function that takes frames from each video clip of the chosen classes and returns the indices, resized frames, and video storage locations after iterating over all the categories mentioned in the array. Now the dataset can be used for training and testing.

5.1.1.4 Split dataset:

Shuffle and split data to create training and testing sets.This negotiates the bias data and also represents the general spread of the data.

Thus it gives a basic idea of similarity and dissimilarity of dataset.

Training data=75%

Testing data=25%

5.1.2 Models:

5.1.2.1 CNN+LSTM:

The CNN LSTM model is a very popular model used often in Machine Learning and Natural Language Processing with common use cases ranging from Speech Recognition to Activity Recognition.

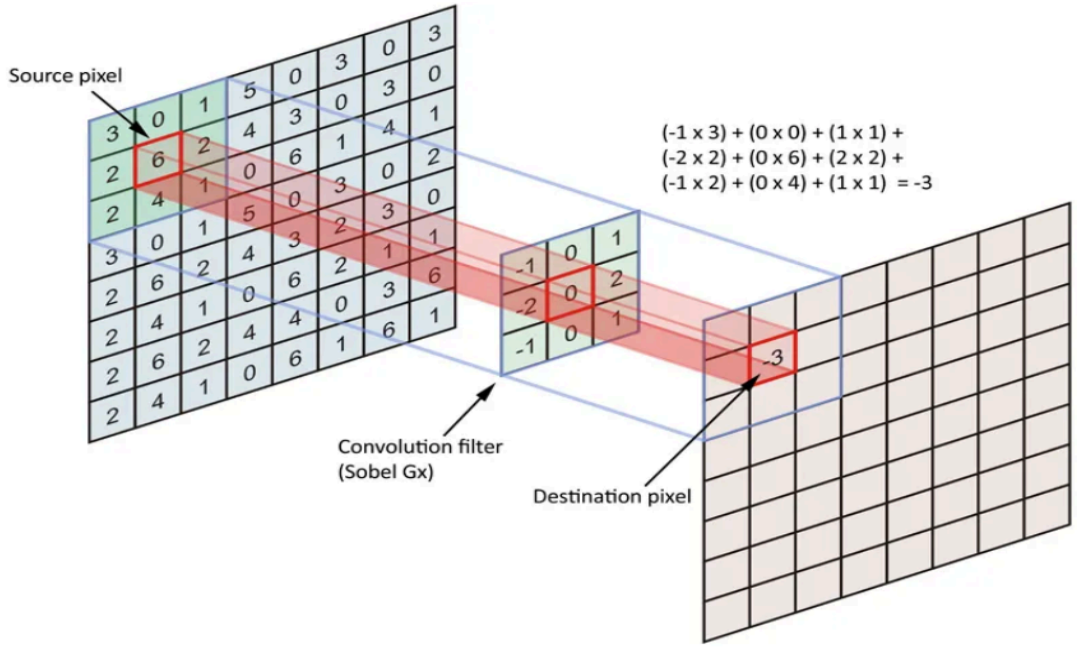

Figure 5.2: Convolution of an image with one filter

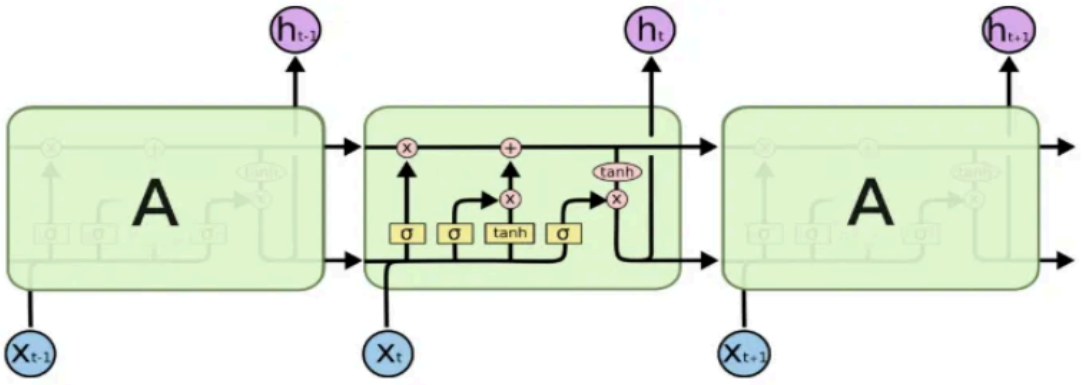

Figure 5.3: A LSTM Cell

The traditional LSTM approach is not suited for spatial inputs and hence the CNN model is preferred in combination with LSTM for sequence classification[29]. The simple task performed by the model is to label images for problems involving either spatial input like pixel(2d), text(1d) or temporal output[30]. The sequential model can be identified as a sequence or summation of two models i.e the Convolutional Neural Network and Long Short

Term Memory. The CNN acts as a front end and is responsible for feature extraction after which lies the LSTM and dense layer that gives the output.

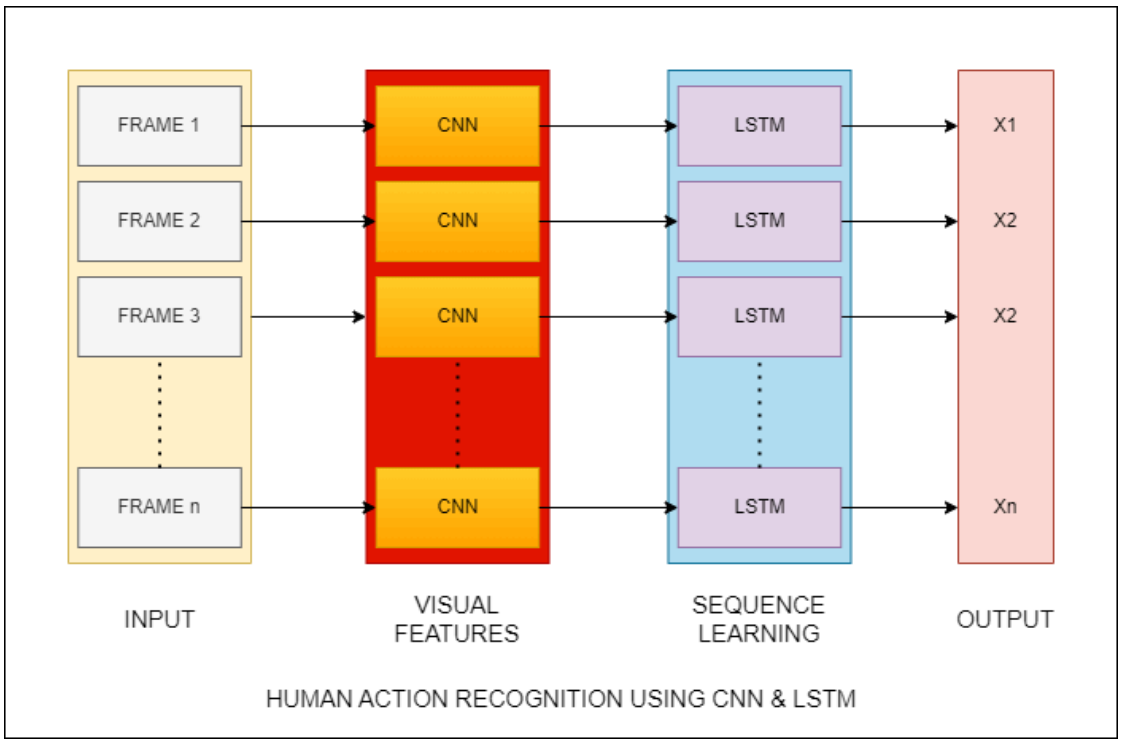

Figure 5.4: HAR using CNN LSTM layers

The CNN model consists of two layers as the following- CNN 2Dand MaxPooling2D. The CNN 2D works on the pixels of the frames obtained from videos while the pooling layers drop their dimensions. This has the limitation of modeling only the given image to its vector representation and not a sequence of pixels which throws light on the role of the LSTM model[31]. The input shape for the independent convolutional model is (Sequence Length, Height, Number of Channels) i.e (20,64,3) obeying its compatibility of expected parameters as 2.

For all the models we define the following parameters:

S- Sequence Length

H- Image height after resizing

W- Image depth after resizing

N- Number of channels

Consider a video of 300 frames(F), and calculate ‘n’ as n=F/S. Therefore every nth frame in the video is considered for training the model.

5.1.2.2 ConvLSTM2D:

You may think of the Convoluted LSTM(2D) model as an enhanced version of the original LSTM models. It makes it possible to segregate the Spatio-temporal characteristics of the provided data. While spatial data are being retrieved, time-dependent features can be derived thanks to the convolution-integrated design[32]. The classification of videos requires this model. We extract the temporal relationships between the distinct frames after obtaining spatial-features from the isolated frames first[33]. The developed model exhibits the capacity to accept 3D input due to the convolution layers embedded. The input shape has the following dimensions: (S, W, H, N) i.e (20, 64, 64, 3). A straightforward LSTM model, on the other hand, would only accept one dimension, making it irreconcilable to model data with properties associated with both space and time.

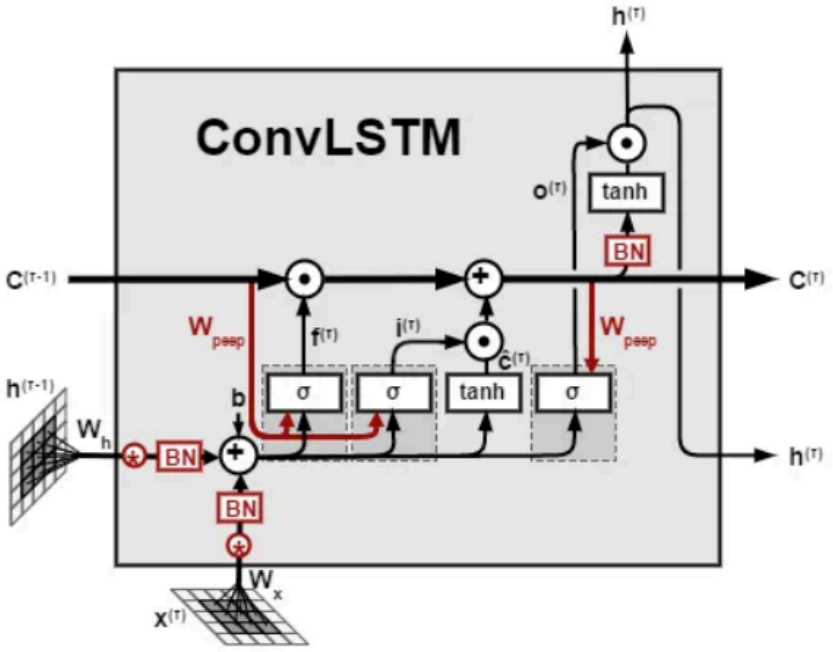

Figure 5.5: A ConvLSTM Cell

a) MaxPooling3D- The primary function of MaxPooling layers are used for dimensionality reduction. Here, the frames taken as input are downsampled. It saves considerable computations cost and improves the efficiency of the model by not taking into consideration avoidable computations. This layer is responsible to downsample the input along its height, depth, and width. It takes into consideration the maximum sequence length for each channel of the input. The strides are currently set to (2,2) along each dimension. Now they are used to shift windows[34].

b) Dropout layers prevent the mode from overfitting on the data. Overfitting is a modeling error in statistics caused as a function is too closely aligned to a limited set of data points[35].The architecture can be varied depending upon how many parameters(2D-Height,Width or 3D-Height,Width, Depth) are we deciding to train with.This is because we may vary the lengths of the subset of the dataset. We now shift the window by strides along each dimension[35].

c) Softmax Classification:

In the ConvLSTM2D layer, the model is fed with the number of filters (3,3)(variable). We also define a kernel size which is used for feature extraction. The output from the above layers is passed as input for the DenseNet layer. This layer uses the flattened output for the learning process of the model despite the inefficiency[33]. Here, the softmax is activated and thus outputs the probability of each action category. Nonetheless, hinge loss provides the margin. Probabilities are considerably simpler for us to interpret as individuals than margin scores (similar to hinge loss and squared hinge loss)[35].

The model has convolution network for input as well as recurrent transformation, and unlike the traditional LSTM model, allows working with spatio-temporal data as input and do the classification. The architecture has only one type of layer and hence is easy to construct with 4 such layers with 4, 8, 16, and 32 filters[33].

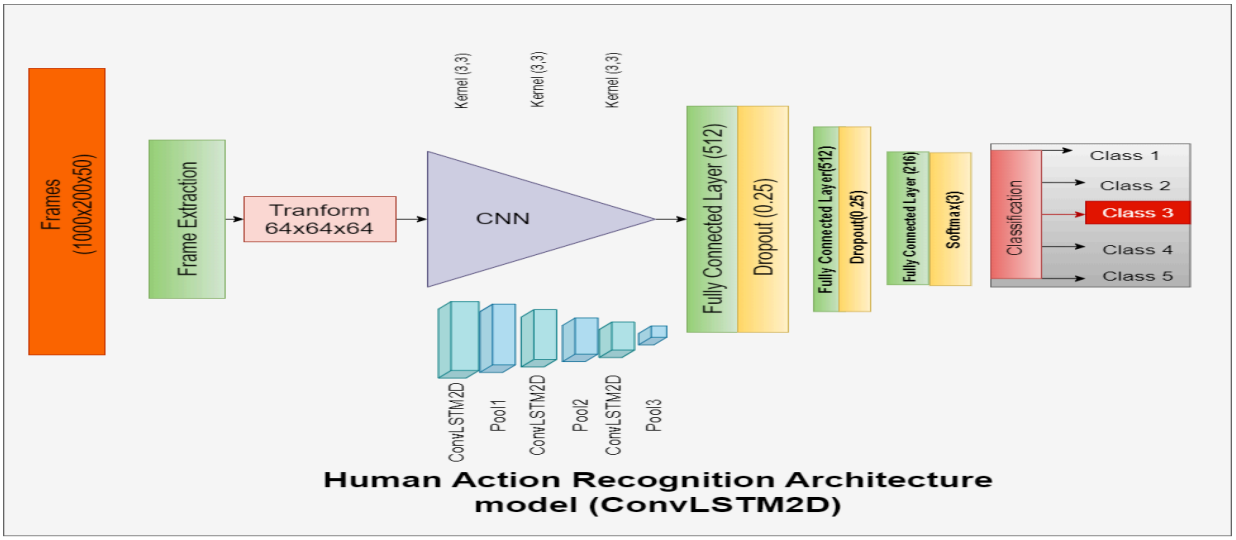

Figure 5.6: Human Action Recognition Architecture model

5.1.2.3 Time distributed Convolutional Network:

In the LRCN Approach we combine Convolution i.e the CNN(2 dimensional) and LSTM layers in an unique/solitary robust model. In the traditional approach we first train the CNN

model for feature extraction and then distinctly use LSTM for image classification. Here too, the Convolutional Neural Networks layers are used for feature extraction. The input frames(resized) are modeled by their height, width to identify spatial features. These features are fed to the LSTM layer. This way the model learns extracted spatiotemporal features directly and creates a robust model with end-to-end training[36].

Conv2D layers are first settled. They are then followed by **MaxPooling2D** and **Dropout** layers.

Flatten layers: These layers are used to flatten the feature extracted from the **Conv2D** layers, before feeding to an **LSTM**.

The **Dense** layer: The input to this layer is the output of the previous LSTM layer. Dense layer specifically has activated Softmax which makes use of these data to produce results or in other words predict the action being depicted in the image or video[37].

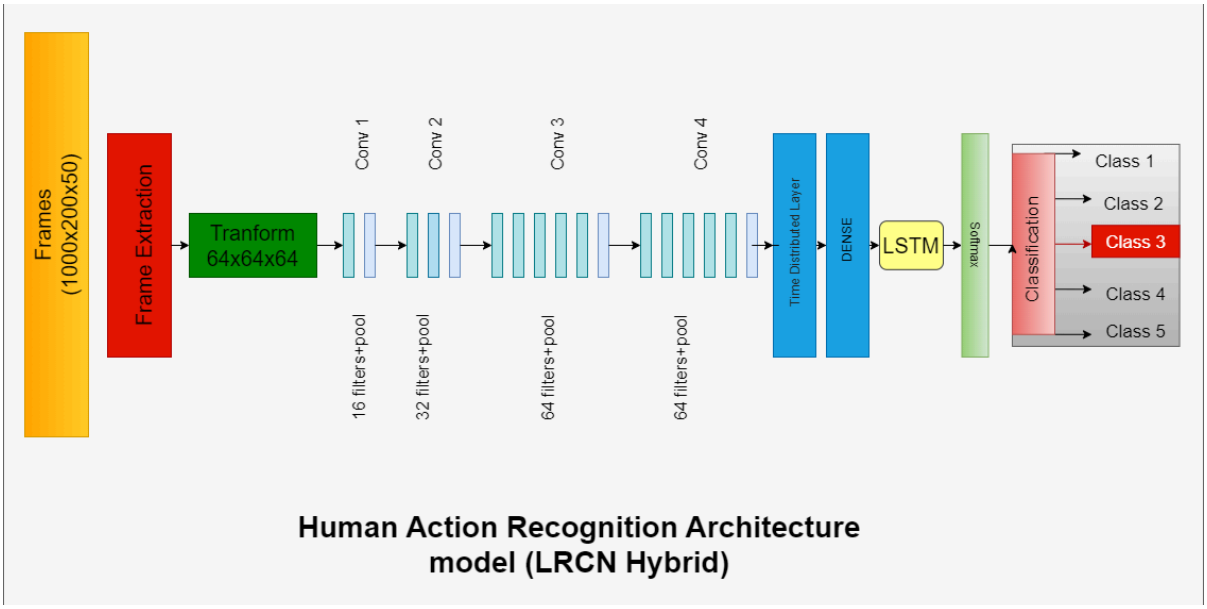

Figure 5.7: LRCN Model

TimeDistributed layer: With the help of this wrapper, you may add a layer to each temporal slice of an input. Each input must have a minimum of three dimensions, with the temporal dimension being the index-one dimension of the first input. In an independent manner, it makes a wrapped layer that takes input as the following (S, W, H, N). In the traditional approach, we had taken the following input-(W, H, N) which is not as efficient as it does not allow the input of the entire video sequence in a unified way.

After the initial resizing and input shape of the form (20,64,64,3) four convolutional layers interpret the pixels. These layers have the following number of filters- 16, 32, 64, and 64. The output obtained is provided as an input to a time-distributed dense layer followed by the LSTM layer, with Softmax activation[36].

The model can be easily built owing to its end to end design and it is because of this efficiency that allows it to be used on time varying spatial data in a plethora of Computer Vision tasks. Compared to the simple LSTM model that has only one dimensional input shape, the LRCN model takes two more inputs other than sequence length. Since it is a sequential model[38], it takes input as a sequence of frames(20 frames) f1, f2, f3, f4…f20 and for each time step t1, t2, t3, t4…t20 classify the frame into one of the 50 action categories. Averaging the predictions, the final classification is done by activated softmax.

5.1.2.4 Swift-Spatio Flow:

a) Feature Extraction using 3D CNN: The Convolutional LSTM designs incorporate a convolutional recurrent cell in an LSTM layer to combine time series processing and CV. The settings for the 3-Dimensional-Convolutional Neural Network is initialized as (64*64*64), with a sequence length of 20. It consists of 3 Convolutional layers as the frontend and a Convoluted LSTM layer in the back for sequential inputs modeling. The first iteration constitutes of a three dimensional CNN with 32 filters. We define a kernel of the setting (3*3*3). A 3D max-pooling layer is added next. The second and third Convolutional iteration resembles the first one but has 32*2 (or 1*64) and 128 filters respectively. However, laying the fourth layer, it contains 512(2*256) filters employed, and the kernel size is set to (2×2×2) which increases the training time of the model significantly. Hence 3 layers with filters 64,128 and 256 filters respectively are suggested as an alternative. However it provides deeper representations of spatio-temporal regions and boosts the accuracy significantly. The pooling layers factor the dimensions by and dropout layers reduces the involvement of the convolutional layers by an extent of 0.25. Set recurrent dropout to 0.2 and the tanh activation function was sufficient in the case.

The final Conv3D layer can be replaced by a Transposed Convolutional layer without changing the parameters like strides and padding. As a result the dimensions of the input frame can be increased depending upon the need. This technique is called upsampling. Also, compared to 2D CNN and other models, the extraction space and time sequence information is much more deep. We wrap this convolutional network with the variant of LSTM layer i.e

ConvLSTM allowing to design a unique model which has better structure than the traditional 3D CNN model or 2D/3D LRCN.

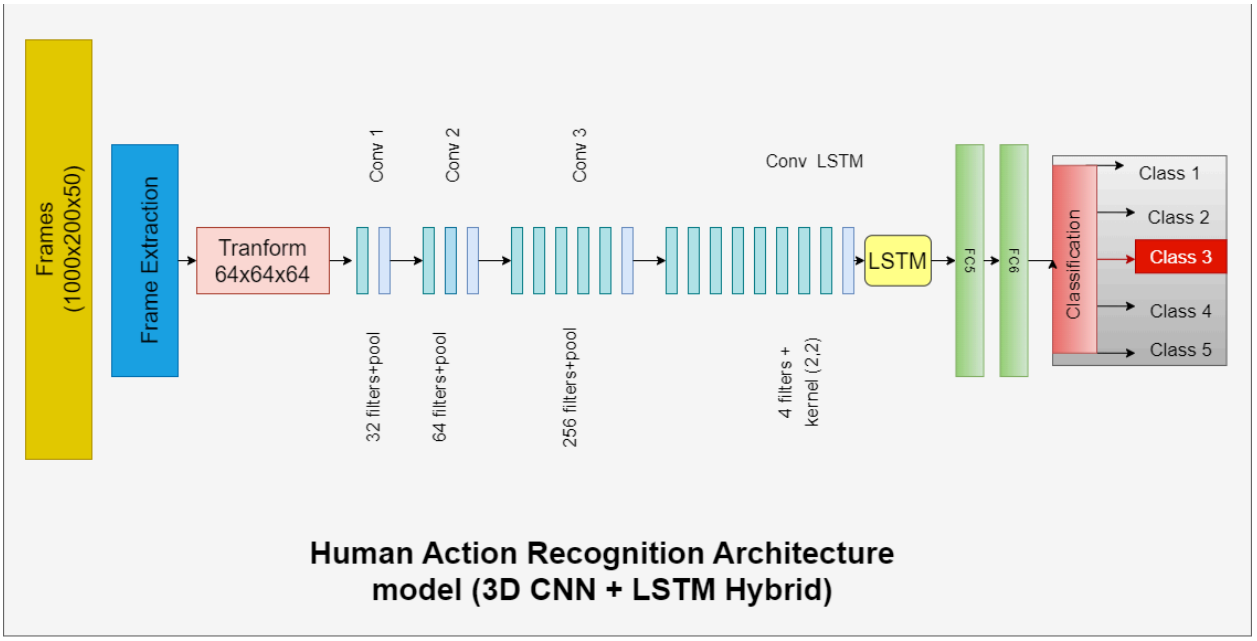

Figure 5.8: Swift-Spatio FlowArchitecture model

b) Impact of LSTM with 3D CNN: The model holds a Long Short Term Memory network and two Fully Connected layers. The spatial parameters/properties to be fed as input to the former layer is determined by the sequence length and frames extracted from the video. Initially we train the model to assess various functions(values) LSTM layer settings in relation to the training set from the UCF-50 dataset. The Fully Connected layer (FC layer) accurately classifies the action sequence in a video. The FC dense layers are deployed after the LSTM layer with softmax activation.

Unlike all the previous models where the kernel moves in only two directions, the kernel of this 3D filter model moves in three directions. A large amount of data is required for training the three dimensional convolutional networks and insufficiency in the training set results in inefficiency in the performance of the model and poor accuracy. Hence datasets of a large number of action classes like the Kinetics or UCF 100 are required. However, this can be avoided by using the LSTM layer with the 3D Convolution for the task of prediction or classification because of their deep nature. This unique design is vital because although SVM

used with 3D CNN is good for Regression with CNN, the same cannot be said for Classification problems. The model is designed such that it best trains under 50 epochs.

Overall the model has a complex architecture with deep layers and can face compatibility issues with expected and found input shape if appropriate parameters are not defined. If designed correctly, it is very robust and accurate at classifying a sequence of three dimensional inputs.

5.2 Code for the project:

5.2.1 Preprocessing and data visualization Code:

```
%%capture
!pip install tensorflow opencv-contrib-python youtube-dl moviepy pydot
# Import the required libraries.
import os
import cv2
import pafy
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf
from collections import deque
import matplotlib.pyplot as plt

!pip install imageio==2.4.1
%matplotlib inline

from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
```

```
seed_constant = 27
np.random.seed(seed_constant)
random.seed(seed_constant)
tf.random.set_seed(seed_constant)
# Create a Matplotlib figure and specify the size of the figure.
plt.figure(figsize = (20, 20))

# Get the names of all classes/categories in UCF50.
all_classes_names = os.listdir('UCF50')

# Generate a list of 20 random values. The values will be between 0-50,
# where 50 is the total number of class in the dataset.
random_range = random.sample(range(len(all_classes_names)), 20)

# Iterating through all the generated random values.
for counter, random_index in enumerate(random_range, 1):

    # Retrieve a Class Name using the Random Index.
    selected_class_Name = all_classes_names[random_index]

    # Retrieve the list of all the video files present in the randomly selected Class Directory.
    video_files_names_list = os.listdir(f'UCF50/{selected_class_Name}')

     # Randomly select a video file from the list retrieved from the randomly selected Class
Directory.
    selected_video_file_name = random.choice(video_files_names_list)

    # Initialize a VideoCapture object to read from the video File.
          video_reader=
cv2.VideoCapture(f'UCF50/{selected_class_Name}/{selected_video_file_name}')

    # Read the first frame of the video file.
    _, bgr_frame = video_reader.read()

    # Release the VideoCapture object.
    video_reader.release()
```

```
    # Convert the frame from BGR into RGB format.
    rgb_frame = cv2.cvtColor(bgr_frame, cv2.COLOR_BGR2RGB)

    # Write the class name on the video frame.
                        cv2.putText(rgb_frame,  selected_class_Name,  (10,  30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

    # Display the frame.
    plt.subplot(5, 4, counter);plt.imshow(rgb_frame);plt.axis('off')



# Specify the height and width to which each video frame will be resized in our dataset.
IMAGE_HEIGHT , IMAGE_WIDTH, IMAGE_DEPTH = 64, 64, 64
sample_shape=(SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH,3)
# Specify the number of frames of a video that will be fed to the model as one sequence.
SEQUENCE_LENGTH = 20

# Specify the directory containing the UCF50 dataset.
DATASET_DIR = "UCF50"

# Specify the list containing the names of the classes used for training. Feel free to choose
any set of classes.
CLASSES_LIST = [
"WalkingWithDog", "TaiChi", "Swing", "HorseRace"]

def frames_extraction(video_path):
    '''
     This function will extract the required frames from a video after resizing and normalizing
them.
    Args:
        video_path: The path of the video in the disk, whose frames are to be extracted.
    Returns:
        frames_list: A list containing the resized and normalized frames of the video.
    '''
```

```
    # Declare a list to store video frames.
    frames_list = []

    # Read the Video File using the VideoCapture object.
    video_reader = cv2.VideoCapture(video_path)

    # Get the total number of frames in the video.
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    # Calculate the interval after which frames will be added to the list.
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)

    # Iterate through the Video Frames.
    for frame_counter in range(SEQUENCE_LENGTH):

        # Set the current frame position of the video.
                        video_reader.set(cv2.CAP_PROP_POS_FRAMES,    frame_counter    *
skip_frames_window)

        # Reading the frame from the video.
        success, frame = video_reader.read()

        # Check if Video frame is not successfully read then break the loop
        if not success:
            break

        # Resize the Frame to fixed height and width.
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

         # Normalize the resized frame by dividing it with 255 so that each pixel value then lies
between 0 and 1
        normalized_frame = resized_frame / 255

        # Append the normalized frame into the frames list
        frames_list.append(normalized_frame)
```

```
    # Release the VideoCapture object.
    video_reader.release()

    # Return the frames list.
    return frames_list

def create_dataset():
    '''
    This function will extract the data of the selected classes and create the required dataset.
    Returns:
        features:          A list containing the extracted frames of the videos.
        labels:            A list containing the indexes of the classes associated with the videos.
        video_files_paths: A list containing the paths of the videos in the disk.
    '''

    # Declared Empty Lists to store the features, labels and video file path values.
    features = []
    labels = []
    video_files_paths = []

    # Iterating through all the classes mentioned in the classes list
    for class_index, class_name in enumerate(CLASSES_LIST):

        # Display the name of the class whose data is being extracted.
        print(f'Extracting Data of Class: {class_name}')

        # Get the list of video files present in the specific class name directory.
        files_list = os.listdir(os.path.join(DATASET_DIR, class_name))

        # Iterate through all the files present in the files list.
        for file_name in files_list:
            # Get the complete video path.
            video_file_path = os.path.join(DATASET_DIR, class_name, file_name)
            # Extract the frames of the video file.
            frames = frames_extraction(video_file_path)
```

```
            # Check if the extracted frames are equal to the SEQUENCE_LENGTH specified
above.
            # So ignore the vides having frames less than the SEQUENCE_LENGTH.
            if len(frames) == SEQUENCE_LENGTH:

                # Append the data to their respective lists.
                features.append(frames)
                labels.append(class_index)
                video_files_paths.append(video_file_path)

    # Converting the list to numpy arrays
    features = np.asarray(features)
    labels = np.array(labels)

    # Return the frames, class index, and video file path.
    return features, labels, video_files_paths



# Create the dataset.
features, labels, video_files_paths = create_dataset()

# Using Keras's to_categorical method to convert labels into one-hot-encoded vectors
one_hot_encoded_labels = to_categorical(labels)

# Split the Data into Train ( 75% ) and Test Set ( 25% ).
features_train, features_test, labels_train, labels_test = train_test_split(features,
one_hot_encoded_labels,
                                                            test_size = 0.25, shuffle = True,
                                                            random_state = seed_constant)
```

5.2.2 Model Architecture Code:

```
def create_CNN3D_model():
    global model
    model = Sequential()

    # Define the Model Architecture.

    # Layer 1- Conv3D
    model.add(Conv3D(64, (3, 3, 3), activation="relu",name="conv1",
                     input_shape=(20,64,64, 3),
                     strides=(1, 1, 1), padding="same"))
          model.add(MaxPooling3D(pool_size=(2,  2,  2),  strides=(1,  2,  2),  name="pool1",
padding="valid"))

    # Layer 2- Conv3D
    model.add(Conv3D(128, (3, 3, 3), activation="relu",name="conv2",
                     strides=(1, 1, 1), padding="same"))
          model.add(MaxPooling3D(pool_size=(2,  2,  2),  strides=(2,  2,  2),  name="pool2",
padding="valid"))

    # Layer 3- Conv3D
    model.add(Conv3D(128, (3, 3, 3), activation="relu",name="conv3",
                     strides=(1, 1, 1), padding="same"))
          model.add(MaxPooling3D(pool_size=(2,  2,  2),  strides=(2,  2,  2),  name="pool3",
padding="valid"))
```

```
 # Layer 4- ConvLSTM2D backend
```

```
    model.add(ConvLSTM2D(filters = 4, kernel_size = (3, 3), activation = 'tanh',data_format = "channels_last",
                         recurrent_dropout=0.2, return_sequences=True, input_shape = (SEQUENCE_LENGTH,
                                                                                      IMAGE_HEIGHT, IMAGE_WIDTH, 3)))

                    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='channels_last'))
    model.add(TimeDistributed(Dropout(0.2)))

    # FC Layer
    model.add(Flatten())

    model.add(Dense(len(CLASSES_LIST), activation = "softmax"))


########################################################################################################

    # Display the models summary.
    model.summary()

    # Return the constructed convlstm model.
    return model
```

5.2.3 Training Code:

```
# Construct the required convlstm model.
model= create_CNN3D_model()

# Display the success message.
print("Model Created Successfully!")

# Plot the structure of the constructed model.
```

```
plot_model(create_CNN3D_model(), to_file = 'create_CNN3D_model.png', show_shapes = True, show_layer_names = True)

 # Create an Instance of Early Stopping Callback
early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience = 10, mode = 'min', restore_best_weights = True)

# Compile the model and specify loss function, optimizer and metrics values to the model
model.compile(loss = 'categorical_crossentropy', optimizer = 'Adam', metrics = ["accuracy"])

# Start training the model.
model_training_history = model.fit(x = features_train, y = labels_train, epochs = 50,
                                   shuffle = True, validation_split = 0.2,
                                   callbacks = [early_stopping_callback])



# Evaluate the trained model.
model_evaluation_history = model.evaluate(features_test, labels_test)



# Get the loss and accuracy from model_evaluation_history.
model_evaluation_loss, model_evaluation_accuracy = model_evaluation_history

# Define the string date format.
# Get the current Date and Time in a DateTime Object.
# Convert the DateTime object to string according to the style mentioned in date_time_format string.
date_time_format = '%Y_%m_%d__%H_%M_%S'
```

```
current_date_time_dt = dt.datetime.now()
current_date_time_string = dt.datetime.strftime(current_date_time_dt, date_time_format)

# Define a useful name for our model to make it easy for us while navigating through multiple saved models.
model_file_name = f'model___Date_Time_{current_date_time_string}___Loss_{model_evaluation_loss}___Accuracy_{model_evaluation_accuracy}.h5'

# Save the Model.
model.save(model_file_name)
```

5.2.4 Testing on videos Code:

```
def predict_on_video(video_file_path, output_file_path, SEQUENCE_LENGTH):

    # Initialize the VideoCapture object to read from the video file.
    video_reader = cv2.VideoCapture(video_file_path)

    # Get the width and height of the video.
    original_video_width = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
    original_video_height = int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))

    # Initialize the VideoWriter Object to store the output video in the disk.
    video_writer = cv2.VideoWriter(output_file_path, cv2.VideoWriter_fourcc('M', 'P', '4', 'V'),
                                   video_reader.get(cv2.CAP_PROP_FPS), (original_video_width,
original_video_height))

    # Declare a queue to store video frames.
    frames_queue = deque(maxlen = SEQUENCE_LENGTH)

    # Initialize a variable to store the predicted action being performed in the video.
    predicted_class_name = ''
   # Iterate until the video is accessed successfully.
  while video_reader.isOpened():
```

```
# Read the frame.
    ok, frame = video_reader.read()

    # Check if the frame is not read properly then break the loop.
    if not ok:
        break

    # Resize the Frame to fixed Dimensions.
    resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

     # Normalize the resized frame by dividing it with 255 so that each pixel value then lies between 0 and 1.
    normalized_frame = resized_frame / 255

    # Appending the pre-processed frame into the frames list.
    frames_queue.append(normalized_frame)

    # Check if the number of frames in the queue are equal to the fixed sequence length.
    if len(frames_queue) == SEQUENCE_LENGTH:

        # Pass the normalized frames to the model and get the predicted probabilities.
        predicted_labels_probabilities = model.predict(np.expand_dims(frames_queue, axis = 0))[0]

        # Get the index of class with the highest probability.
        predicted_label = np.argmax(predicted_labels_probabilities)

        # Get the class name using the retrieved index.
        predicted_class_name = CLASSES_LIST[predicted_label]
# Write predicted class name on top of the frame.
    cv2.putText(frame, predicted_class_name, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    # Write The frame into the disk using the VideoWriter Object.
    video_writer.write(frame)
```

```
# Release the VideoCapture and VideoWriter objects.
```

```
    video_reader.release()
    video_writer.release()


from moviepy.editor import VideoFileClip

from moviepy.video.io.VideoFileClip import VideoFileClip
# Construct the output video path.
output_video_file_path =
f'{test_videos_directory}-Output-SeqLen{SEQUENCE_LENGTH}.mp4'

# Perform Action Recognition on the Test Video.
predict_on_video(input_video_file_path, output_video_file_path, SEQUENCE_LENGTH)

# Display the output video.
VideoFileClip(output_video_file_path,audio=False,target_resolution=(300,None)).ipython_di
splay()
```

5.3 Results

Figure 5.9: Horse Race recognized by Swift-Spatio Flow model

Figure 5.10: Walking with dog recognized by Swift-Spatio Flow model

Figure 5.11: Swing action recognized by Swift-Spatio Flow model

Figure 5.12: Taichi action recognized by Swift-Spatio Flow model

Chapter 6

Conclusion And Future Enhancement

6.1 Comparison:

The proposed model is a novel model in the field of Human Action Recognition and Image Classification in general. The conventional models which use the Convolutional Neural Networks and LSTM layer have dominated the area of Human Action Recognition but it comes with noticeable drawbacks.The proposed model has taken into consideration these points and overcame them which is elucidated in the following manner. Various factors and metrics have been referred to for drawing conclusions like Accuracy, Precision, Architecture Complexity, Computational Cost, Training time, and Size of datasets.

The proposed model is a novel model in the field of Human Action Recognition and Image Classification in general. The conventional models which use the Convolutional Neural Networks and LSTM layer have dominated the area of Human Action Recognition but it comes with noticeable drawbacks.The proposed model has taken into consideration these points and overcame them which is elucidated in the following manner. Various factors and metrics have been referred to for drawing conclusions like Accuracy, Precision, Architecture Complexity, Computational Cost, Training time, and Size of datasets.

Sr. No	**Model**	**Accuracy (in %)**	**Precision Score (in %)**	**Recall Score (in %)**	**F1 Score (in %)**
1.	CNN+LSTM	76.12	75.94	74.17	75.86
2.	ConvLSTM2D	78.95	78.74	76.14	78.68
3.	Time Distributed CNN	88.50	88.00	87.52	87.71
4.	3D CNN (Using UCF 101)	91.65[39]	89.96	90.82	91.1

Sr. No	**Model**	**Accuracy (in %)**	**Precision Score (in %)**	**Recall Score (in %)**	**F1 Score (in %)**
5.	Swift-Spatio Flow (3D CNN+LSTM)	94.89	94.37	93.45	93.56

Table 6.1: Comparison of various metrics for different HAR. Models

Sr. No	Model	Training time(Ranked least to most)	Complexity	Size of dataset required	Computationally expensive
1.	CNN+LSTM	1	Low	Small(50 Action classes)	Low
2.	ConvLSTM2D	4	High	Medium to Large(50-200 Action classes)	High
3.	Time Distributed CNN	3	Very High	Large(100+ Action classes)	Very High
4.	3D CNN	5	High	Huge (400-700 Action classes)	Very High
5.	Swift-Spatio Flow (3D CNN+LSTM)	2	Very High	Medium (50-100 Action classes)	Medium-High

Table 6.2: Comparison of various characteristics for different HAR. Models

The proposed model is a novel model in the field of Human Action Recognition and Image Classification in general. The conventional models which use the Convolutional Neural Networks and LSTM layer have dominated the area of Human Action Recognition but it comes with noticeable drawbacks.The proposed model has taken into consideration these points and overcame them which is elucidated in the following manner. Various factors and metrics have been referred to for drawing conclusions like Accuracy, Precision, Architecture Complexity, Computational Cost, Training time, and Size of datasets.

The proposed model is a novel model in the field of Human Action Recognition and Image Classification in general. The conventional models which use the Convolutional Neural Networks and LSTM layer have dominated the area of Human Action Recognition but it comes with noticeable drawbacks.The proposed model has taken into consideration these points and overcame them which is elucidated in the following manner. Various factors and

metrics have been referred to for drawing conclusions like Accuracy, Precision, Architecture Complexity, Computational Cost, Training time, and Size of datasets.

The ConvLSTM2D approach and the LRCN (LongTerm Recurrent Neural Network) approach could be an easy and go to method for classification problems like Human Action Recognition(HAR) precisely, since it allows classification of spatio-temporal data. While dealing with images, Convolutional Neural Networks(CNN) are preferred i.e for image classification, CNN are best for feature extraction. On the other hand Recurrent Neural Network(RNN) and Long Short Term Memory Network(LSTM) are favored for temporal data. Now, since HAR is based on video classification, the go to approach would be to use a suitable combination of CNN and LSTM because a video consists of a certain number of frames and use them for learning spatial and temporal features respectively.

However the ConvLSTM2D model, despite being a robust variant of the LSTM model, is a complex model and it is difficult to fine tune the hyperparameters because it significantly increases the training time. Another problem is overfitting, and hence the model may not train sufficiently on smaller datasets resulting in a failure to draw meaningful interpretations. This is why the model did not have a convincing accuracy when trained on the UCF 50 dataset. Similarly, the LRCN model despite being a powerful one for spatio-temporal data had similarly disadvantages like being computationally expensive which was obvious because of involving large numbers of parameters. It is difficult to explain the results produced by it since it has a complex architecture. The biggest challenge would be to achieve a high accuracy using unsupervised machine learning techniques and hence the model is preferred only when a labeled dataset is available and although it provides accuracy more than that of ConvLSTM layers, this approach has non negotiable drawbacks.

The complex nature of 3D CNN models allows the extraction of extensive or exhaustive features which might not be drawn out or interpreted from the other two networks. But a big drawback of this method is the requirement of large datasets which automatically results into high computational cost in order to train the deep layers of the network.(ref of third paper from lit review)Although the model can give high accuracy, it depends on collecting large sied related data. The UCF 50 dataset is insufficient to train such a model by a large margin. However, few datasets that can be used in action recognition problems are Kinetics 400, Kinetics 700, UCF 101, and Something-Something datasets. The results of these models can

be easily interpreted and explained but the process of determining the correct architecture is time consuming owing to the complexity in deciding the number of layers. SImilarly, optimizing the model is a challenge too as a small change in configuration of parameters like kernel size, batch size, filters, epochs etc can vastly affect the accuracy and training time of the three dimensional model.

Therefore, amongst the existing models, LRCN model and 3D CNN model are the picked models for 2 dimensional and 3 dimensional networks respectively. The former is preferred if dataset is small and the later is used if dataset is large enough which eventually would have better accuracy if fine tuned with parameters. However as mentioned above, there are considerable drawbacks of the mentioned techniques. This is why the proposed model is an important addition to the field and this is why the novel model is better than the existing models:

1. Has better accuracy than the existing models including the 2D and 3D models. The training time was less than LRCN and ConvLSTM2D model and although the difference is not much, it is a substantial gain as it involves 3D neural network and yet trains quicker on the same dataset.

2. Trains on smaller datasets like UCF 50 while this dataset is insufficient for training a 3D CNN by a fair margin as such a model requires up to 400 action classes. Thus, because of being trained sufficiently on a small dataset, the model is saved from being computationally expensive.

3. Beats 3D CNN trained on other dataset(Kinetics 400) on accuracy by only a sizeable margin but trains much more speedily. Requires less memory to store parameters.

Disadvantage:

1. Complexity of the model increases. The layers have to be deployed in the proper order and proportionate numbers in order to achieve the best performing model.

Hence the time consuming process to define the architecture. However, this does not affect the performance because of using optimal parameters which again requires additional time as it is a calculated trial and error process.

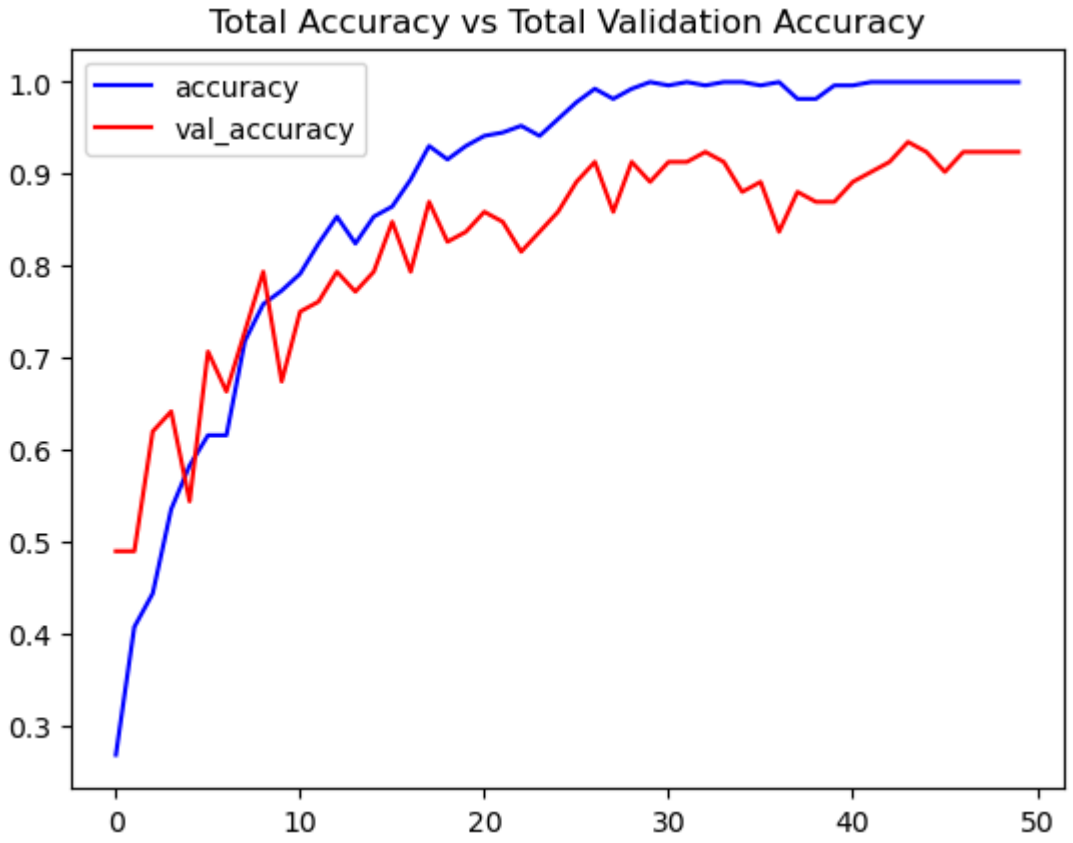

Fig 6.1: Plotting Accuracy for the model

6.2 Future Implementations:

1. Aid for the blind people- Once the model has been successfully deployed, after fulfilling certain demands for accuracy levels, the Hardware Description Language(HDL) model can be integrated with appropriate hardware and text to voice feature in order to assist the blind people by guiding them about the environment around them in real time through instruments like glasses.

2. Driverless(Automated) cars- Similar technique can be used to deploy guides for self driving cars for them to move escaping obstacles through a camera.

3. CCTV surveillance- Unclear images or recordings create ambiguity in opinions or intel extracted from them. This can be solved by integrating the cameras with the HDL model.

References

[1] R. H. Khan, I. Ullah, and F. Ullah, "Human Action Recognition: A Review," arXiv preprint arXiv:1909.11489, 2019.

[2] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, "Towards Good Practices for Very Deep Two-Stream ConvNets," arXiv preprint arXiv:1507.02159, 2015.

[3] H. Gao, Z. Zhang, L. Ji, J. Zhang, and X. Tian, "A survey on deep learning for human action recognition," Neurocomputing, vol. 334, pp. 139-153, 2019.

[4] M. Shafiee, A. Mohammadi, and H. F. Pourreza, "Deep Learning for Human Action Recognition: A Review," arXiv preprint arXiv:2103.04030, 2021.

[5] Ravi, D., Wong, C., & Lo, B. (2016). Wearable sensors for monitoring of physical and cognitive performance: a review of recent developments. Frontiers in public health, 4, 289.

[6] Wijnen, F. (2014). Recognizing complex activities: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(9), 1814-1838.

[7] Gormley, M. R., & Lucey, P. (2016). A survey of salient feature and deep learning approaches for action and activity recognition in video. Intelligent Multimedia Surveillance, 101-129.

[8] P. Zhang, J. Guo, and F. Liu, "Human action recognition: A survey," IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 9, pp. 2660-2679, Sep. 2019.

[9] J. Chen, X. Lian, and H. Yin, "Recent Advances in Human Action Recognition Using Depth Sensors: A Review," IEEE Access, vol. 9, pp. 145114-145129, 2021.

[10] S. Zhang, L. Wu, and R. Cui, "A Survey of Natural Language Processing for Smart Home," IEEE Access, vol. 8, pp. 40256-40270, 2020.

[11] Y. Gao, X. Li, and X. Luo, "Towards Exploring and Exploiting the Inductive Bias of Transformer for Visual Question Answering," IEEE Trans. Multimedia, vol. 23, no. 5, pp. 1685-1697, May 2021.

[12] X. Wang, W. Li, W. Liang, T. Cao, and H. Zhang, "Towards Real-time Object Detection and Tracking in Multi-camera Systems," IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 1, pp. 181-194, Jan. 2021.

[13] Serpush, F., & Rezaei, M. (2020). Complex Human Action Recognition in Live Videos Using Hybrid FR-DL Method. *arXiv*.

[14] Wang, G., Guo, Y., Wong, Y., & Kankanhalli, M. (2022). Distance Matters in Human-Object Interaction Detection. *arXiv*.

[15] Hara, K., Kataoka, H., & Satoh, Y. (2017). Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?. *arXiv*.

[16] A. Khan, W. Han, J. Kim and J. Sohn, "Human activity recognition using smartphone sensors and deep convolutional neural networks," in Sensors, vol. 19, no. 5, p. 1230, 2019, doi: 10.3390/s19051230.

[17] K. Hara, H. Kataoka and Y. Satoh, "Learning Spatiotemporal Features with 3D Convolutional Networks," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 4489-4497, doi: 10.1109/ICCV.2017.480.

[18] UCF Center for Research, "CRCV," *Ucf.edu*. [Online].

[19] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A Large Video Database for Human Motion Recognition. ICCV, 2011.[Online]
Available

[20] B. G. Fabian Caba Heilbron Victor Escorcia and J. C. Niebles, 'ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding', in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 961–970.

[21] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, 'A Short Note on the Kinetics-700 Human Action Dataset', CoRR, vol. abs/1907.06987, 2019.

[22] M. Firman, 'RGBD Datasets: Past, Present and Future', CoRR, vol. abs/1604.00999, 2016.

[23] B. Jia, Y. Chen, S. Huang, Y. Zhu, and S.-C. Zhu, 'LEMMA: A Multiview Dataset for Learning Multi-agent Multi-view Activities', in Proceedings of the European Conference on Computer Vision (ECCV), 2020.

[24] K. Lai, L. Bo, and D. Fox, 'Unsupervised feature learning for 3D scene labeling', 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 3050–3057, 2014.

[25] J. Wang, Z. Liu, Y. Wu, and J. Yuan, 'Mining actionlet ensemble for action recognition with depth cameras', in 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1290–1297.

[26] Chollet, F. (2018). Deep learning with Python. Manning Publications.

[27] Chollet, F. et al. (2015). Keras: The Python Deep Learning library. arXiv preprint arXiv:1502.01852.

[28] Abadi, M. et al. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI).

[29] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," arXiv preprint arXiv:1409.1259, 2014.

[30] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315-323.
[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.
[32] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
[33] Shi, X., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Advances in neural information processing systems (pp. 802-810).
[34] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. arXiv preprint arXiv:1603.04467.
[35] Chollet, F. (2015). Keras. GitHub repository. Retrieved from https://github.com/fchollet/keras
[36] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2625-2634, 2015.
[37]Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," Proceedings of the IEEE International Conference on Computer Vision, pp. 19-27, 2015.
[38]Y. Li, X. Zhang, and D. Chen, "End-to-end joint learning of motion and depth for optical flow estimation using convolutional neural networks," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6344-6352, 2018.
[39]J. Lee, Y. Lee, and J. Kim, "Human Activity Recognition Using 3D CNNs and Convolutional LSTM," in 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, Dec. 2018, pp. 144-149, doi: 10.1109/ICARCV.2018.8581269.