Skateboard Trick Recognition through an AI-based Approach

Kris Saliba

Supervisor: Dr. Joseph Bonello

June 2024

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Information Technology (Honours) (Software Development).

L-Università ta' Malta
Faculty of Information &
Communication Technology

Abstract

Acknowledgements

Contents

List of Figures

List of Tables

1 Introduction

Skateboarding dates back to the late 1940s or early 1950s, evolving from a leisure activity for surfers on flat land to a worldwide phenomenon [**SkateboardingEncyclobeadia**]. This worldwide appeal was further amplified by its inclusion as an official sport in the 2020 Tokyo Olympic Games [**SkatebaordingOlimpics**]. This event had a significant impact on the sports popularity globally, but particularly evident in the United States, as demonstrated by the spike in the number U.S. skateboarding participants between the years 2020 and 2021, as illustrated in Figure 1.1.

This dynamic sport encompasses various disciplines and riding styles, each offering unique challenges for skateboarders to explore. Two of the most prominent styles are "vert" and "street." Vert skateboarding involves riding on specialised obstacles, namely, half-pipes and ramps, emphasising aerial manoeuvres. Street skateboarding takes place in urban environments, utilising various obstacles that can be found outdoors, including stairs, rails, ledges, gaps or flat ground for skaters to showcase their creativity [**skateStyles**].

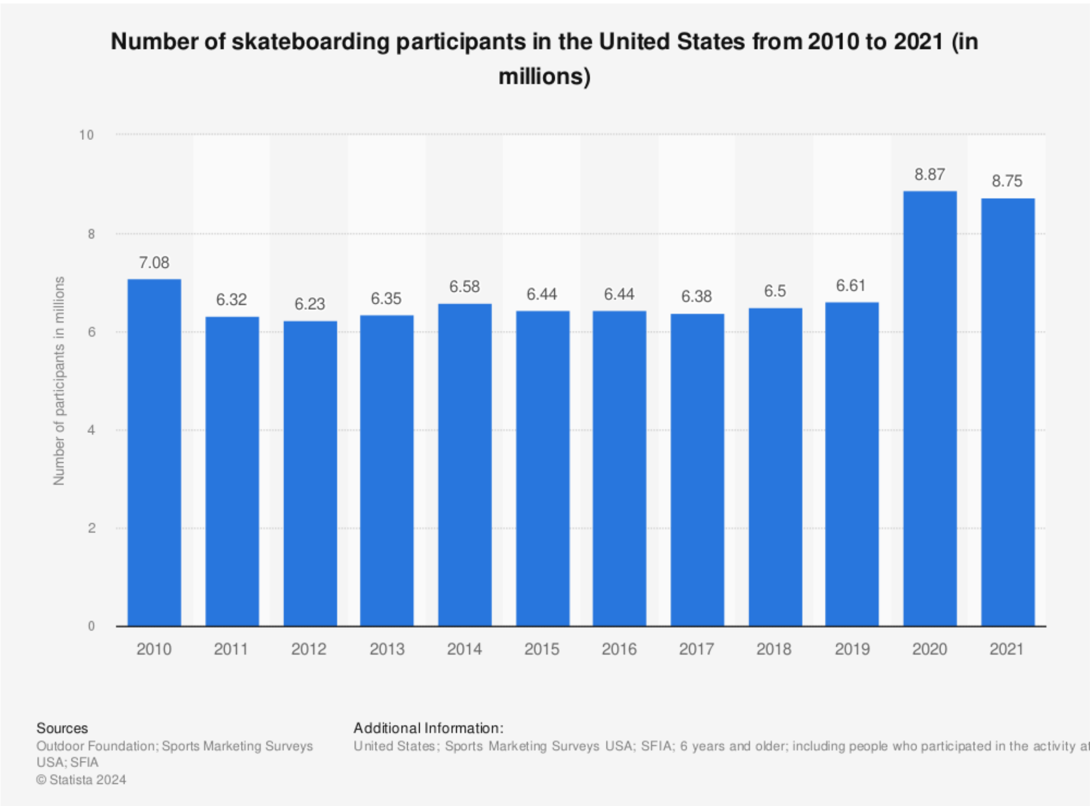


Figure 1.1 Number of skateboarding participants in the United States from 2010 to 2021 in millions. Reproduced from [**skatestatistics**], data sourced from Outdoor Foundation [**outdoorfounation**].

1.1 Motivation

The recent surge in skateboarding's popularity across digital platforms like YouTube and Instagram has increased the demand for on-screen trick identification , enabling inexperienced viewers to identify tricks performed by skaters in videos through a digital overlay that labels each manoeuvre. Traditional methods require manually labelling each trick in a video, a process that is not only time consuming but also error prone. Automating this process could save time in this regard, but also offer real-time benefits for live streamed content, such as skateboard competitions and events, providing insightful footage to viewers, further enhancing the understanding and appreciation for the sport. An Artificial Intelligence (AI) model capable of accurately classifying skateboard tricks could significantly improve viewer experience in these scenarios.

Despite the popularity in computer vision, particularly in video activity recognition, limited research exists in the domain of skateboard trick classification. While some initial studies such as, Shapiee et al. (2020) [**skatePaper1**] and Hanxiao Chen (2023) [**SkateboardAIPaper**] explored this domain, there remains a gap in research, driving the need for further growth in this area.

This task poses significant challenges, due to diverse camera angles, lighting conditions and complex skateboard movements. However, this research aims to overcome these challenges and create a robust and accurate tool that can reliably classify skateboard tricks from video.

1.2 Hypothesis

The hypothesis for this research asserts that employing a combination of deep learning strategies and preprocessing techniques can improve the accuracy and robustness in classifying skateboard tricks.

1.3 Research Questions

- How effectively can combining deep learning techniques and algorithms accurately classify skateboard tricks from video data?
- What is the impact of different video pre-processing techniques on classification accuracy?

1.4 Aims and Objectives

1.4.1 Aims

- To classify skateboard tricks using images extracted from videos into their respective classes.
- To compare the performance of Deep Learning architectures in the context of skateboard trick classification.

1.4.2 Objectives

- To Implement a set of Deep learning architectures and evaluate them using appropriate metrics.
- Employ suitable frame processing techniques
- Augment and preprocess the images to determine any improvement on performance.

1.5 Structure

This study is structured systematically to explore the potential of deep learning in skateboard trick recognition. Chapter 2 begins by establishing the necessary background knowledge required for this study, while Chapter 3 provides an overview of the past research conducted in this domain.

Next, Chapter 4 outlines the approach behind the artefact, with implementation specifics provided in Chapter 5. Chapter 6, discusses the outcomes and evaluates them against other studies and finally, Chapter 7 presents the final findings and identified limitations that this research encountered.

2 Background

2.1 Skateboard Tricks

Skateboard tricks can be described as dynamic manoeuvres that involve complex coordination of the skateboard and the skateboarder's body. Skateboard tricks can be categorised into several types, including rotations, grinds, performed on ledges or rails and manuals, where the skater balances on two wheels. The key to successfully performing these tricks is appropriate foot placement, which is critical for controlling the skateboard's speed and direction. This control allows skateboarders to manipulate the board in ways that replicate specific tricks, showcasing not only their technical abilities and creativity.

- **Ollie:** One of the first tricks beginners learn. Where the skateboarder pops the tail of the board while simultaneously sliding their foot across the nose of the board, causing the board to level out in the air, used to jump over obstacles.
- **Kickflip:** A trick where the skateboarder flips the board under their feet while jumping, making it spin 360°around the x-axis.
- **Pop-Shuvit:** A trick where the skateboarder scoops the board with their back foot causing a 180°rotation around the y-axis.

Skateboarders continually innovate and come up with new trick combinations, contributing to the dynamic nature of the sport.

2.2 Machine Learning

Machine Learning (ML) can be defined as a field of study that explores algorithms and statistical models employed by computer systems to execute tasks without the need to be explicitly programmed. It is particularly applicable in situations where the information we seek from a dataset is not interpretable, and as the volume of available datasets continues to surge so does the demand for machine learning [**ML_Algorithms**].

Morris (2019) [**UnderstandingLSTM**] characterises ML as the advancement of algorithms that progressively enhance their performance through practice, suggesting that the more training the learning algorithm undergoes, the better it becomes at executing tasks. Numerous critical factors shape a model's performance within this phase, as exemplified by Budach et al. (2022) [**TheEffectsofDataQualityonMachineLearningPerformance**]. Such factors include dataset quality and diversity, data preprocessing, the selection of a suitable model architecture, training time and the fine-tuning of hyper-parameters.

There exist three main categories for ML models [**ML_Algorithms**]:

- **Supervised:** This is a ML concept that involves training a model to make classifications based on input data that has been labelled with the correct label.
- **Unsupervised:** This ML concept concentrates on discovering relationships within data when there are no predefined ”correct” answers or labelled examples to guide the learning process. These models are left to autonomously explore and divulge structures in the data.
- **Reinforcement:** This type of learning consists of an agent that interacts with the environment and learns from the continuous feedback it receives in the form of rewards or punishment.

2.3 Activity Recognition

Activity recognition is the process of identifying and categorizing human activities from video sequences. Human activities involve a wide range of motions and interactions with objects, varying from simple isolated actions like dancing to more complex activities that engage multiple body parts and external objects such as a football match. The human ability to perceive these behaviours is a trivial task; yet, it is a challenging problem for computers due to the sequential nature and the resemblance of visual content in such activities [**ActionRecognitionDeepBi-DirectionalLSTM**, **AReviewOfHumanActivityRecognitionMethods**].

2.4 Neural Networks

2.4.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a class of Machine Learning models that are inspired by the interconnected systems of neurons found in the nervous system of living organisms. They consist of connected nodes capable of learning from their environment and adapting to complex patterns in data [**FundamentalsOfNeuralNetworks**]. Figure 2.1 depicts a schematic representation of an ANN. The diagram is organised into three fundamental layers: the Input Layer, the Hidden Layer(s) and the Output Layer [**FundamentalsOfArtificialNeuralNetworksAndDeepLearning**].

- **Input Layer:** This is the set of neurons that serve as the initial entry point for external data. Each input neuron in this layer corresponds to a specific feature or variable used in the Neural Network model.
- **Hidden Layer(s):** This is the set of neurons that are located between the Input and Output Layers where the network captures complete non-linear behaviours of data and feature transformations.
- **Output Layer:** This is the set of neurons that provide the final predictions produced by the Neural Network. Depending on how the ANN is configured, the final output can be continuous, binary, ordinal, or count.

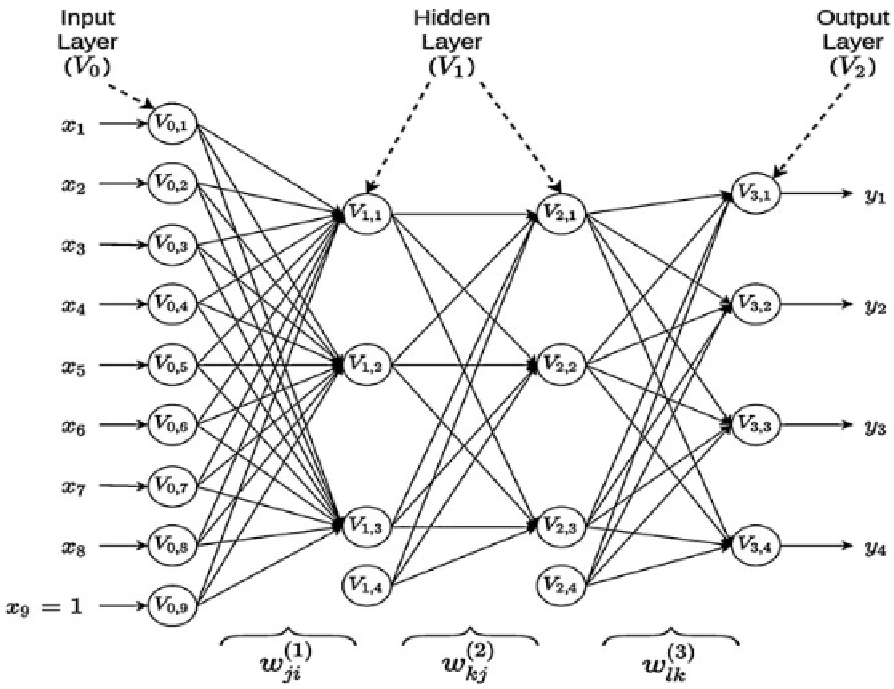


Figure 2.1 Schematic Representation of an Artificial Neural Network. Reproduced from López et al. (2022) [**FundamentalsOfArtificialNeuralNetworksAndDeepLearning**]

2.4.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), as described by Gu et al. (2018) [**RecentAadvancesInConvolutionalNeuralNetworks**] are a category of Deep learning architectures with roots in the biological visual perception mechanisms of living organisms. These networks have gained widespread attention for their incredible performance in the field of image recognition and pattern recognition tasks.

CNNs are comprised of three types of layers: convolutional layers, pooling layers and fully-connected layers. The convolutional layer applies a set of filters (or kernels) that slide across the input data performing a localised dot product between their weights and the corresponding values of the input data. The results are summed up to generate a single value in the feature map. The pooling layer applies an aggregation funciton such as max pooling or average pooling to create a downsamplaed representation of the input data are. Finally, the fully-connected attempt to transform the outputs from the previous layer into an output vector that represents a score corresponding to a class label.

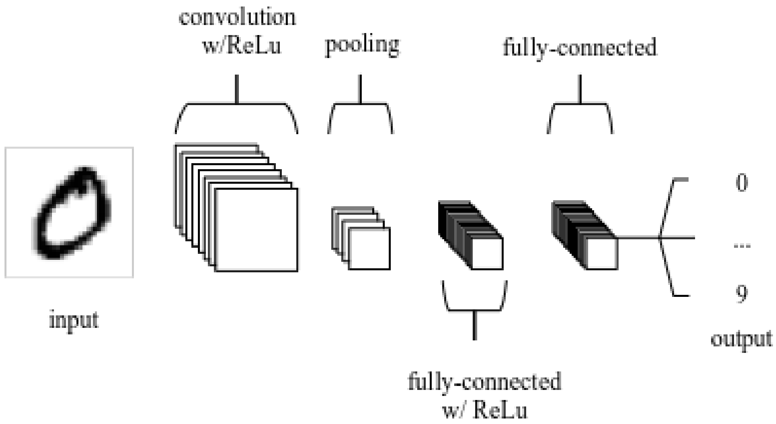


Figure 2.2 CNN architecture comprising 5 layers. Reproduced from O'Shea and Nash (2015) [**introductiontoCNNs**]

Figure 2.2 showcases a simplified CNN architecture comprising of 5 layers, Nonethelss, the complexity of CNNs can be scaled by stacking multiple layers, thereby increasing the network's depth to cater for more complex tasks.

2.4.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a subset of Neural networks that are designed for sequential data processing. RNNs are capable of modelling dynamic relationships in sequential data by feeding signals from past time steps back into the network. However, they are limited due to their inability to access long-term data, limited to approximately ten sequential time steps.This constraint arises from the challenge of dealing with vanishing or exploding gradients in the backpropagation procedure while processing extended sequences, as discussed in prior works [**UnderstandingLSTM**, **Long-termConvolutionalNeuralNetworksforVisualRecogntion**].

Long Short-Term Memory Networks

To address the limitation that RNNs encounter in capturing long-term dependencies, Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs) were introduced as an extention to RNNs in 1997 [**learningPriceseTimingWithLSTM**, **originalLSTMPaper**]. LSTM networks inherently extend the RNNs memory enabling them to campture dependencies across more than 1,000 time steps depending on the network's complexity. These models are also able to tackle the vanishing problem associated with RNNS and offer more biologically credible solution [**UnderstandingLSTM**].

These types of networks consist of three gates: input, output and forget gates. The input gate, determines whether new information will be uploaded to the network, the output gate manages whether current cell values contribute to the output, and the forget gate determines whether existing information will be preserved or removed [**theperformanceoflstmandbilstminforcastingtimeseries**]. Figure 2.3 illustrates the architecture of an LSTM network, displaying the interaction between the cell state and the various other gates that regulate the flow of information through the network.

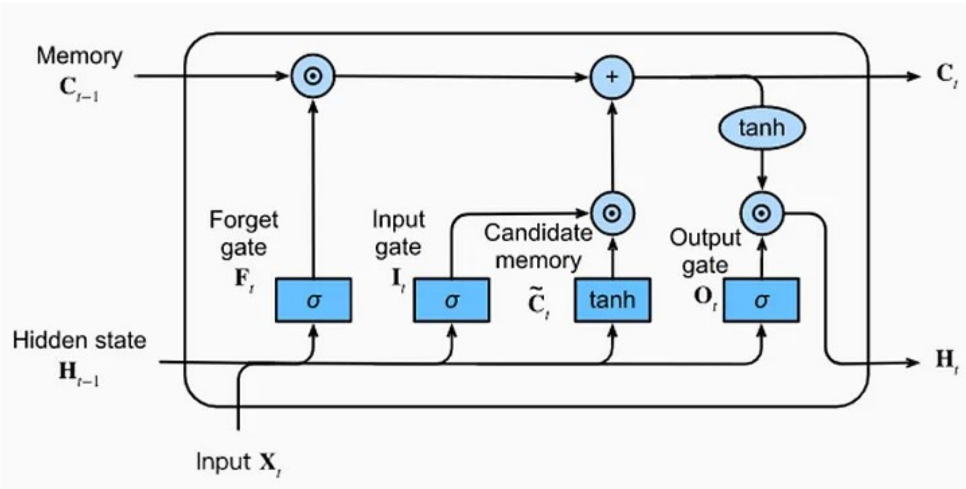


Figure 2.3 LSTM architecture. Reproduced from ....

Bidirectional LSTMs

Bidirectional LSTMs (BiLSTMs) are sophisticated variants of LSTMs designed to ehnance the ability to capture patterns in time series data and improve learning long-term dependencies. Unlike LSTMs that only process data in a single direction (from past to future), BiLSTMs utilise a more intuitive approach by applying two LSTM models to the input data. In the first round, the LSTM is applied to the input data, and in the second round, it is applied to the reversed input sequence. Furthermore, Tavakoli et al. (2019) [**theperformanceoflstmandbilstminforcastingtimeseries**], concluded that BiLSTMs reported better accuracies compared to regular LSTMs.

Figure 2.4 demonstrates the structure of a BiLSTM network, showcasing the flow of data between the two parallel LSTM layers. The first LSTM processes the input data in the forward direction, from x_1 to x_6, while the other LSTM in the backwards direction from x_6 to x_1, allowing the network to learn from past and future contexts. The forward LSTM units are connected through weights Wf_1 and Wf_2, while the backward LSTM

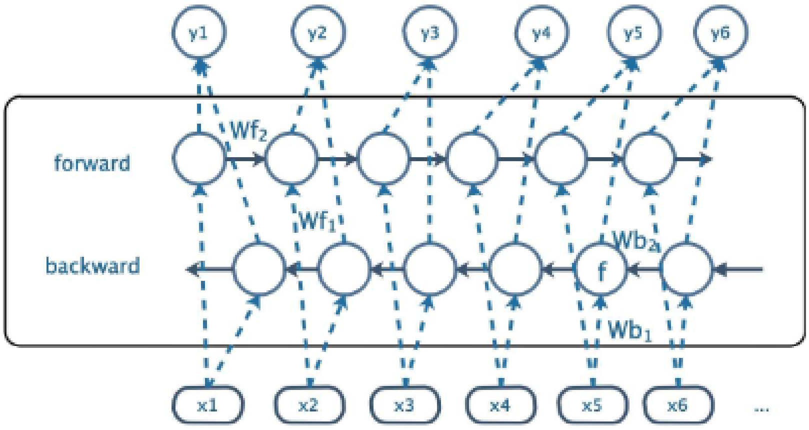


Figure 2.4 BiLSTM Structure. Reproduced from Du et al. (2020) [**du2020power**]

units are connected through weights Wb_1 and Wb_2. By learning from sequences in both directions, BiLSTMs are particularly effective in scenarios where past and future contexts are crucial.

3 Literature Review

3.1 Activity Recognition

Building on the foundational understanding of activity recognition (AR) defined in the background, it's important to acknowledge the impact it has on various sectors. The recent advancements in recognising human actions in videos have not only revolutionalised sectors such as healthcare and security as demonstrated in the studies [**3DhumanActionDetectionForHealthCareSystems**], [**HumanActivityRecognitionSecurityAndMonitoring**], but also hold extensive potential in the realm of sports.

The research conducted by K. Host and M. Ivašić-Kos in [**HARinSportsComputerVision**] along with Wu et al. in [**Asurveyonvideoactionrecognitioninsports:Datasetsmethodsandapplicatio** further elaborate on this potential, such as its numerous applications in categorising complex sports actions, injury prevention methods and refinement of game strategies through video analysis. These studies also propose various methodological advancements, including the utilisation of Deep Learning models to enhance accuracy, the exploration of multimodal data sources for sports activities and an emphasis on real-time analysis capabilities. These insights provide valuable techniques that can be leveraged for the development of AR models in the field of sports.

The study by Beddiar et al. [**VisionBasedHARASurvey**], identifies two main streams of Human-Computer Interaction (HCI) technologies: Contact-based and Vision-based systems. The authors categorise contact-based HCI as those technologies that require physical user interaction through mediums such as accelerometers, wearable sensors and multi-touch interfaces. Alternatively, the authors describe vision-based methods as the simplification of HCI due to more natural human communication, eliminating the need for physical contact or equipment. These methods use image and video data to recognise human activities offering an advantage in terms of societal acceptance and usability.

3.1.1 Challenges in this field

The domain of AR comes with many challenges as described by Zhang et al. (2017) [**ReviewOfHumanActivityRecognitionUsingVisionBasedMethod**]. The researchers suggest that while certain scenarios utilise static cameras such as surveillance systems, most situations that benefit from AR adopt dynamic recording devices such as mobile phones and sports event broadcasts. These dynamic devices introduce a significant level of complexity as a result of their tricky dynamic backgrounds present in the video footage. Zhang et al. also point out specific difficulties posed by long-distance and low-quality videos, often encountered in environments like crowded public spaces and sports events. The camera distance, results in smaller subjects, making detailed analysis of human movements more challenging while lower-quality videos further complicate the task for Human Activity Recognition (HAR) systems

The challenges highlighted by Zhang et al. [**ReviewOfHumanActivityRecognitionUsingVision** in the domain of AR are directly relevant to the development of a skateboard trick classifier. In scenarios like televised skateboarding events, skaters may appear relatively small to accommodate the entire skatepark, This factor, along with the complexity of the background as a result of dynamic recording, poses a challenge for trick recognition in live broadcasts. Furthermore, if a skateboard trick classifier is intended for personal development, then it may encounter videos of lower quality, again complicating the task of trick recognition. Addressing these challenges is crucial for the development of a skateboard trick classifier that performs well in real-world conditions.

3.1.2 Preprocessing techniques

Preprocessing is a crucial step in the development of ML models, especially in the field of AR. It involves the application of various techniques to enhance raw data before applying the Machine Learning algorithms.

Data Augmentation

As a result of the limited research on the emergence of a skateboard trick classifier, there is a significant lack of open-source datasets featuring skateboard tricks. This lack of data presents an opportunity to employ data augmentation techniques, especially valuable in studies with limited data. Methods such as flipping, rotating, scaling and colour manipulation not only artificially enhance the size of the original dataset but also lower the likelihood of overfitting as highlighted in prior works, [**DataAugmentationCanImproveRobustness**, **AnOverviewOfOverfittingAndItsSolutions**]. In the realm of Skateboard trick classifiers, Shapiee et al (2020) [**skatePaper1**] effectively employed data augmentation techniques to expand their dataset, demonstrating the application of these methods in improving model performance for trick classification.

Feature Extraction

Having optimised the distribution of pixel values through the normalisation process, the next crucial step is feature extraction. Xudong Jiang [**FeatureExtractionForImageRecognitionAndComputerVision**], describes this technique as the process of capturing the core attributes of an object through the elimination of redundancies, resulting in a set of numerical features ideal for classification. CNNs excel at this due to their capabilities in detecting complex patterns and enhancing features. While they are primarily used in image recognition, as an initial step before deploying classification algorithms, their effectiveness in feature extraction is demonstrated by studies like Manjunath Jogin et al. (2018) [**FeatureExtractionUsingCNNandDeepLearning**] where they achieved 86% accuracy using CNNs for feature extraction alongside several classifiers. Beyond this primary function, CNN's ability to extract complex features from images makes them very effective when coupled with other models for more complex tasks such as AR. In the context of skateboard trick classification, CNNs can efficiently extract detailed spatial features like board rotations, foot positioning and limb movements. Such features can then be passed through a sequence analysis model like an LSTM, which have been shown to be efficient at capturing temporal information.

Transfer Learning

Following the discussion on CNNs for feature extraction, it is important to highlight the role of transfer learning in enhancing ML models, especially in fields with limited data like skateboard trick recognition. The concept of transfer learning as detailed by the studies by Weiss et al. (2016) [**ASurveyOfTransferLearning**] and Soni et al. (2018) [**ApplicationAndAnalysisOfTransferLearningSurvey**] involves using a pre-trained ML model to leverage its experience for a new, but related task. Furthermore, the study by Sargano et al. (2017) [**HARUsingTransferLearningWithDeepRepresentations**], employed transfer learning with pre-trained deep CNNs like AlexNet [**AlexNetCite**] and GoogleNet [**GoogleNetCite**], for HAR. Notably, they utilised these models for feature extraction, followed by an SVM classifier for final classification. Their approach showcases the resource-efficient and time-saving nature of transfer learning, evident in their impressive accuracies of 98.15% and 91.47%. Moreover, research on transfer learning is not unique within the field of skateboard trick classification. Two other studies [**skatePaper1**], [**SkateboardAIPaper**] have utilised this approach and achieved high accuracies. However, a more in-depth analysis of these papers can be found in the section 3.2. These studies strongly suggest exploring the use of various pre-trained models for feature extraction in the development of a skateboard trick classifier.

3.1.3 Activity Recognition Techniques

The accurate classification of skateboard tricks poses a unique challenge for computer vision due to the sport's dynamic and complex nature, characterised by rapid movements, potential occlusions and camera angles. This section explores various computer vision techniques and architectures employed in previous literature, exploring their potential impact on this specific task.

Deep Learning (DL) techniques have become increasingly popular over traditional ML methods, for their ability to learn feature representations automatically from raw data, significantly improving performance [**AReviewOnComputerVisionBasedMethodsForHARRec** One particularly effective DL architecture is the CNN-LSTM. This approach leverages the CNNs strength in extracting spatial features from individual frames and LSTMs ability to capture temporal information across frames, leading to a deeper understanding of video content.

The study by Orozco et al. (2020) [**HARRecognitionInVideosUsingARobustCNNLSTMAppro** adopted this approach to test its effectiveness against three AR datasets: KTH, UCF-11, HMDB-51, particularly focusing on how the number of LSTM units impacted performance. The authors employed transfer learning, utilising the VGG16 model [**VGG**] for feature extraction from videos, followed by an LSTM network for classification. Their findings show that 360 LSTM units achieved an accuracy of 93.86% on the KTH dataset, while 320 units led to an accuracy of 91.93%. However, the performance on the HMDB-51 dataset dropped, with 400 LSTM units resulting in a lower accuracy of 47.36%. These findings show the potential of the CNN-LSTM approach, particularly for simpler datasets, highlighting the need for further investigation and optimisation for more complex datasets like HMDB-51.

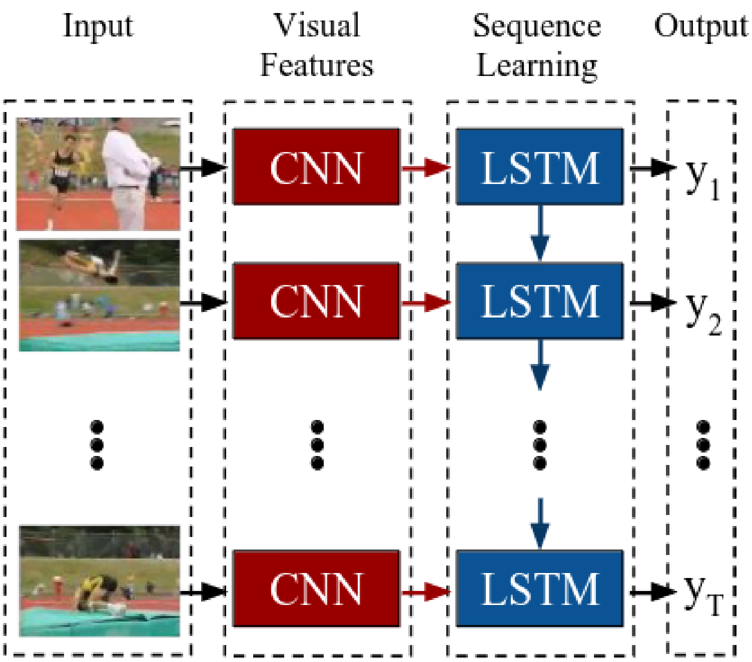


Figure 3.1 CNN-LSTM Architecture. Reproduced from Donahue et al. (2015) [**LongTermRecurrentConvolutionalNetworksForVisualRecognitionAndDescription**]

Building upon the foundational CNN-LSTM architecture, a recent study by Saoudi et al. (2023) [**Advancinghumanactionrecognition:ahybridapproachusingattention-basedLSTMand**
enhances this model by incorporating three key advancements:

1. **3D Convolutional Neural Networks (3D CNNs):** Unlike regular CNNs, data is processed as 3D volumes, enabling them to capture both spatial and temporal information by performing convolution operations on all three dimensions: width, height and time. The authors opted to use the I3D model, leveraging transfer learning to bypass the resources required to train one from scratch. The I3D model was selected for its proven efficiency and its ability to be fine-tuned for activity recognition tasks.

2. **Bi-directional Long-Short-Term Memory (BiLSTM) network:** The authors chose to use a variant of LSTM called Bi-directional Long Short-Term Memory (BiLSTM). Whereas LSTMs only process data in one direction, BiLSTMs can analyse data in both directions, allowing them to understand relationships between preceding and subsequent actions.

3. **Attention Mechanisms:** Saoudi et al. incorporated attention mechanisms after their BiLSTM, enabling the model to prioritise particular parts of the input data. This technique allowed the model to learn which temporal features were most applicable to the task, providing a more detailed representation of the input and ultimately, improved performance.

With the integration of these advancements, Saoudi et al. achieved model accuracies of 97.98% and 96.83% on the HMDB51 and UFC101 datasets respectively, demonstrating the potential of 3D CNNs, BiLSTMSs and attention mechanisms for activity recognition tasks.

3.2 Advancements in Skateboard Trick Classification

In the emergent field of skateboard trick classification, leveraging activity recognition techniques from a video have led to two primary methodologies among researchers. The first technique involves utilising signals obtained from skateboard-mounted accelerometers or signals artificially generated based on the findings of prior studies. These signals are then fed into a study-dependent model for classification, as outlined by Abdullah et al. (2021) [**skateboardClassificationTransferLearningPipelinesAccelermetry**] and Corrêa et al. (2017) [**skateboardTrickClassifierUsingAccelerometryAndML**]. The second approach employs computer vision techniques, leveraging video footage of skateboard tricks to train and refine models for accurate trick identification, as depicted by the studies Shapiee et al. (2020) [**skatePaper1**] and Hanciao Chen (2023) [**SkateboardAIPaper**].

3.2.1 Accelerometer-based approach

The study by Corrêa et al. (2017) [**skateboardTrickClassifierUsingAccelerometryAndML**], obtained their sample data by artificially generating 543 signals based on prior research, utilising MATLAB 2015 and Signal Processing Toolbox. These signals were then categorised into five distinct classes representing different skateboard tricks, each with various samples ranging from 30 to 50 per class, across three axes (X, Y and Z). This study developed and validated individual Artificial Neural Networks (ANNs) for each axis, as well as the combination of the three: ANN XYZ, displaying the potential of Neural Networks to categorise multidimensional skateboard tricks. The ANNs are all multilayer feed-forward neural networks (MFFNNs), structured into three distinct layers. They feature an input layer with 82 neurons, a hidden layer, comprised of 23 neurons utilising a tan-sigmoid transfer function and an output layer consisting of 5 neurons with a softmax function. Finally, the study achieved high accuracies, with ANNs X, Y and Z achieving 94.8%, 96.7% and 98.7%, respectively, while the combined ANN XYZ achieved an accuracy of 92.8%.

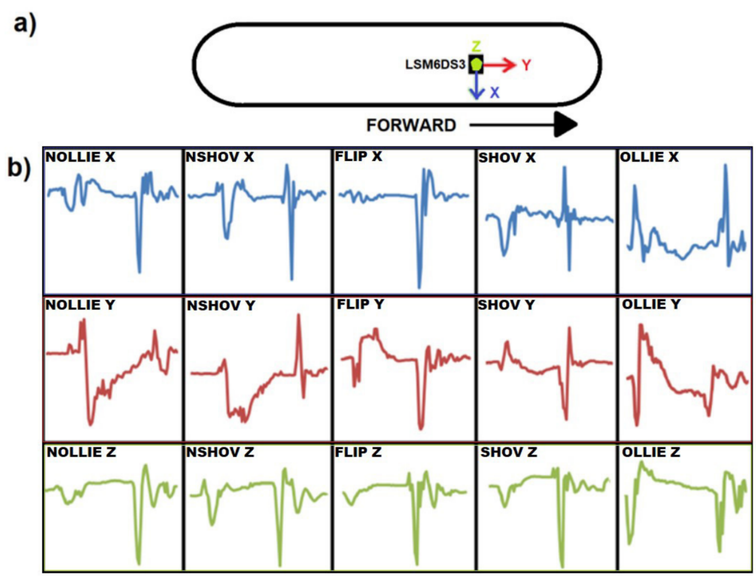


Figure 3.2 add caption

3.2.2 Computer Vision-based Approaches

The study by Shapiee et al. (2020) [**skatePaper1**] leverages a custom data set comprising videos capturing the execution of five distinct skateboard tricks, each attempted five times. Each video spans two to three seconds, yielding a total of 750 images by extracting 30 frames per video. This study made use of data augmentation techniques to expand their dataset further. Consequently, they introduced an additional 2,250 images, achieving 3,000 images in their data set. On the other hand, Chen (2023) [**SkateboardAIPaper**] compiled a comprehensive data set by collecting videos from multiple platforms, including YouTube, Twitter and Instagram. Furthermore, Chen trained the model using 15 fundamental tricks commonly observed in competitive settings, illustrated in Figure 3.3. The researcher collected 50 videos per trick, summing up to a total number of 750 videos. Of these, 45 videos per trick were allocated for training, and the remaining 5 were reserved for validation.

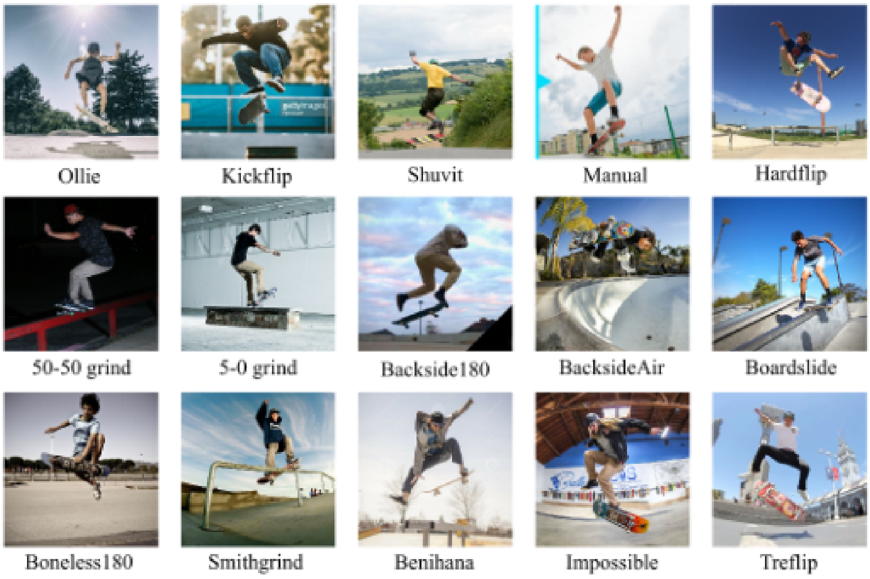


Figure 3.3 Tricks selected for Chen's models. Reproduced from Chen (2023) [**SkateboardAIPaper**]

In the abstract by Hanciao Chen [**SkateboardAIPaper**], extensive experimentation is conducted using diverse models, exploring various combinations of CNN-LSTM and CNN-BiLSTM architectures. The study also incorporated attention mechanisms and explored transfer-based methods for activity recognition. This study further documents and analyses important metrics such as training time, training accuracy and validation accuracy for each model experimented on. Among these, the top three models that stood out in terms of validation accuracy were the ResNet50 with Attention and BiLSTM (84%), ResNet50 with BiLSTM (81%) and ResNet50 with LSTM (80%). Chen's study provides valuable insight into the application of diverse models in activity recognition in skateboarding.

4 Methodology

4.1 Class Establishment

This study pursued a multi-class classification strategy, targeting three fundamental skateboard tricks: ollie, pop-shuvit and kickflip. These tricks were selected based on two primary criteria. Firstly, they are often associated with the first tricks learnt by beginners, highlighting their role in foundational skateboarding skills. Secondly, their popularity within the skateboarding community, often performed in competitions emphasises their relevance, making them highly relevant for analysing and evaluating competitive performance.

4.2 Data Preparation

4.2.1 Dataset

This study utilised video recordings of skateboarders performing tricks as its primary data source. To ensure model robustness, the final dataset consisted of videos across diverse environmental conditions and varying skateboarder skill levels.

The initial dataset was sourced from the publicly available ”SkateboardML” repository on GitHub [**lightningdrop2020skateboardml**], comprising 200 video clips corresponding to two common tricks: the ollie and the kickflip. To expand the dataset's diversity, additional data was obtained through direct communication with Hanxiao Chen, the author of the SkateboardAI paper [**SkateboardAIPaper**]. This communication yielded a dataset containing 750 videos covering 15 distinct tricks. Given the wide range of tricks included in this dataset, many were beyond the scope of this study, therefore this study only encorporated a subset of the this data in its final dataset. The final dataset included 128 videos per class, totalling 384 videos.

4.2.2 Labelling techniques

This study explores two primary labelling techniques: the folder-based and the text-based approach as illustrated in Figures 4.1 and 4.2. The folder-based method categorises videos into folders named after their corresponding class label offering a simple organisation method. On the other hand, the text-based approach, lists each video's path and corresponding label in a text file, providing more flexibility. Given the limited number of classes and manageable dataset size, this study selected the folder-based approach, considering the extra complexity of the text-based method unnecessary for this project.

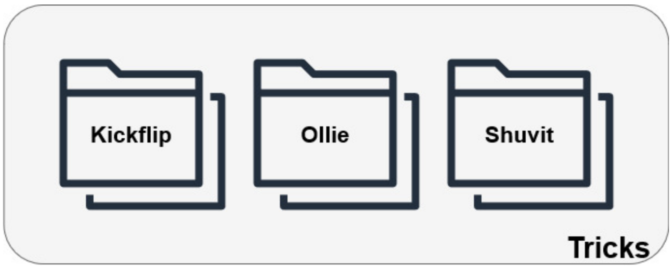


Figure 4.1 Folder-based labelling.

Figure 4.2 Text-based labelling

4.3 Preprocessing

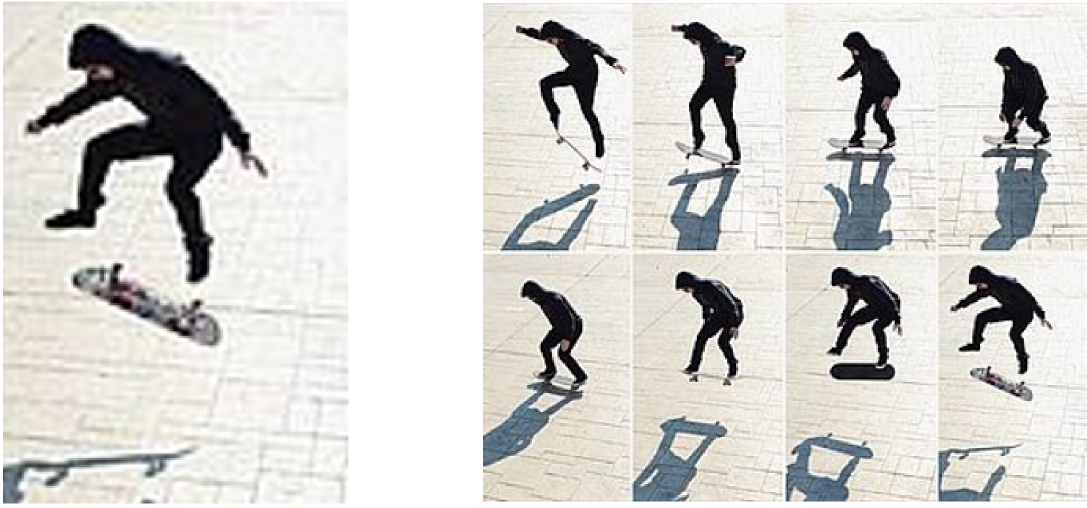
4.3.1 Frame Extraction

Recognising complex human actions demands the examination of sequential data as opposed to relying on single frames or images [**ActionRecognitionDeepBi-DirectionalLSTM**]. To illustrate this point, consider the example of a skateboarder executing a challenging trick like the "Kickflip", as depicted in Figure 4.3. By feeding an AI model a single image of this trick as shown in Figure 4.3a, it may misinterpret the manoeuvre as another trick due to its subjective nature. Whereas, by providing the model with a sequence of frames, as illustrated in Figure 4.3b, the entire trick sequence is captured including the wind-up, the trick and the landing, capturing actions such as foot placement and board rotation, collectively define the skateboard trick.

This study explored two primary frame extraction techniques: uniform sampling and optical flow method. The former technique evenly splits a video into its corresponding frames by using a pre-calculated step size determined by the number of frames of a video. Wherease second technique leverages optical flow, a method that assigns weights to frames determined by the motion of pixels between them. With this approach, frames with the greatest weightings are chosen for extraction, resulting in capturing frames with the most movement. Figure 4.4 illustrates a visualisation of this technique being applied to a skate trick.

4.3.2 Data Augmentation

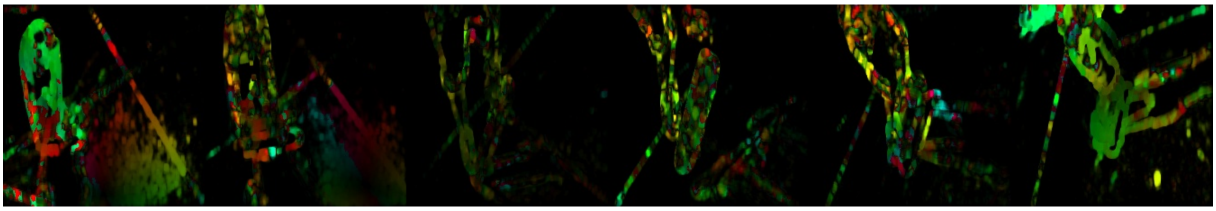
The limited number of data available for this research necessitated the implementaiton of data augmentation techniques to enlarge the dataset's size before model training.



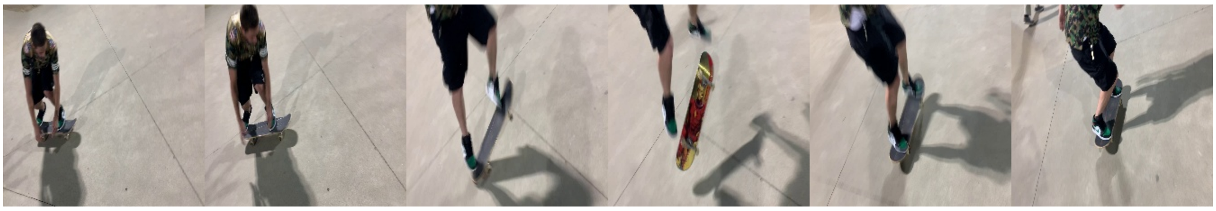
(a) A Single Frame of a ”Kickflip”.

(b) Multiple Frames of a ”Kickflip”.

Figure 4.3 Comparison of a Single Frame and Multiple Sequential Frames.



(a) Optical flow visualisation of a kickflip sequence, highlighting regions with significant motion.



(b) Extracted frames from the sequence based on the greatest weightings from the optical flow representation.

Figure 4.4 Visualisation of optical flow: (a) Optical flow representation (b) Individual frames extracted.

Techniques such as rotating, flipping and adding noise to images also served to add more diversity to the dataset mitigating the risk of overfitting. This risk arises from the over-parametrised nature of the chosen models, whose trainable parameter count surpasses the size of the dataset, causing them to be vulnerable to overfitting.

4.3.3 Normalisation

During the preprocessing stage, normalisation played a crucial role in preparing the video frame data before further processing. In particular this study utilised min-max normalisation to scale the pixel intensities between 0 and 1. Normalising data in this manner ensured that the model's input values data was within a standardised range.

4.3.4 Feature Extraction with Transfer Learning

This research leveraged two well-established CNNs for this task: VGG16 and ResNet50. Both models were pre-trained on the large ImageNet dataset [**imageNetDataset**], leveraging extensive image recognition knowledge that were then fine-tuned to the domain of skateboard trick recognition. Table **??** summarises the key characteristics of VGG16 and ResNet50.

VGG16: VGG16, introduced in 2014 by Simonyan and Zisserman [**VGG**], is a CNN architecture known for its success in image classification and feature extraction for Action Recognition tasks. This model is relatively simple made up of 16 convolutional and fully connected (FC) layers with repeated use of 3x3 convolutional filters to preserve spatial information. Furthermore, VGG16 has proved to be effective for feature extraction in previous literature[**SkateboardAIPaper**], supporting its suitability in this study.

Resnet50: Resnet50, introduced in 2016 by He et al. [**Resnet50**], is another powerful CNN architecture that is well known for its advancements in image classification and frame extraction capabilities. Unlike VGG16, this model has a deeper architecture employing 50 convolutional layers allowing it to learn more intricate feature representations of the data. Furthermore, it incorporates residual connections to counteract the vanishing gradient problem that is often caused by many layers.

Model	**VGG16**	**ResNet50**
Published	2014	2016
Layers	16 (Convolutional and FC layers)	50 (Convolutional layers)
Parameters	$\sim$138 million	$\sim$25 million
Input Image Size	224x224	224x224

Table 4.1 Characteristics of Transfer Learning Models VGG16 and ResNet50

The final four fully-connected (FC) layers of these models, which are typically used for classification were removed. This was crucial to adapt these models for feature extraction, as the aim was not to classify each frame of the video, but to extract a collection of features that could help sequence models gain a better understanding. These features in the form of vectors not only encapsulate the visual information present in each frame but also more abstract ideas such as low-level information (like edges/shapes) to higher-level features like objects and even actions themselves.

4.3.5 Dimensionality Reduction

The use of pre-trained models used in this study generate high-dimensional feature vectors. While these features capture valuable information, a large number of dimensions can lead to challenges. Firstly, it increases the computational costs of training models. Secondly, it could potentially lead to the ”curse of dimensionality” as explained by Venkat (2018) [**curseofdimensionality**], where models may struggle to learn effectively with high dimensional data. To address these issues, dimensionality reduction is employed after feature extraction to reduce the number of features while still retaining valuable information for classification.

4.4 Adapted Models

This research investigates the performance of four different deep learning architectures for skateboard trick classification. These models leverage pre-trained CNNs for feature extraction followed by sequence modelling techniques to capture the temporal dynamics of skateboard tricks. The architectures investigated are as follows.

1. **VGG16-LSTM:** This architecture leverages the pre-trained VGG16 model to extract features, which are then connected to an LSTM for sequence modelling.

2. **ResNet50-LSTM:** This architecture utilises the deeper ResNet50 pre-trained CNN for feature extraction, followed by an LSTM for sequence modelling. The increased depth of ResNet50 potentially allows it to learn more complex feature representations than VGG16.

3. **VGG16-BiLSTM:** This architecture employs the VGG16 model for feature extraction and a BiLSTM network for sequence modelling. BiLSTMs are known to be able to learn dependencies in both directions of a sequence, which could be useful for capturing subtle details in skateboard tricks.

4. **ResNet50-BiLSTM:** This architecture combines ResNet50 for feature extraction with a BiLSTM network. This combination leverages the potential advantages of deeper feature learning and the bi-directional capabilities of BiLSTMs.

The effectiveness of these models in prior research for video-based Action Recognition tasks motivated their selection for this study. For instance, Orozco et al. (2020) achieved 91.93% accuracy using a VGG-LSTM architecture [**HARRecognitionInVideosUsingARobustCNNLSTMApproach**]. Additionally, Chen’s work [**SkateboardAIPaper**] explored all four of these models in the context of skateboard trick classification and demonstrated the potential for competitive results. While this study also explored attention mechanisms within these models, the main aim is to build upon existing knowledge and potentially achieve better performance.

4.5 Architecture

The Deep Learning Architecture, as illustrated in Figure 4.5 demonstrates the data flow pipeline for the skatebaord trick classifier. The process begins with preprocessing the labeled video data including frame extraction, augmenting the training set for variability and reducing deminsionality for efficiency. Pre-trained CNNs such as VGG16 and ResNet50, then extract features from the preprocessed frames, before feeding them to their respective sequence models, namely BiLSTM and LSTM.

The accuracy of these models are then evaluated on a held-out test set, to ensure the models performance on unseen data. Finally, the evaluation utilises performance plots such as confusion matrices to visualise classification performance, model epoch loss and accuracy graphs to track training history.

4.6 Evaluation Methods

To thoroughly evaluate the performance of the proposed models, this study employed various evaluation metrics, including accuracy, precision, recall and F1-score. These metrics provided valuable insights into the model's ability to correctly classify different tricks.

The above mentioned metrics depend on the following definitions, described in the context of the "kickflip" class and tabulated in a confusion matrix.

- **True Positive (TP)**: The number of instances that the model correctly identified a video as containing a kickflip.
- **True Negative (TN)**: The number of instances that the model incorrectly identified a video as containing a kickflip.
- **False Positive (FP)**: The number of instances that the model correctly identified a video as not containing a kickflip.
- **False Negative (FN)**: The number of instances that the model incorrectly identified a video as not containing a kickflip.

Accuracy: Measures the proportion of correctly classified instances, over the total number of predictions made by the model. This metric provides a generic indicator of the model's reliability, however this specific metric can be sensitive in scenarios with class imbalance [**classificationAssessmentMethodstharwat2020**].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Precision: Measures the proportion of predicted positive instances that are truly real positives [**Evaluation:fromprecisionrecallandF-measuretoROCinformednessmarkednessandcorrel** In the context of kickflip detection, it represents the proportion of true kickflips against all predicted kickflips.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

Recall: Measures the proportion of positive instances that are truly predicted positive [**Evaluation:fromprecisionrecallandF-measuretoROCinformednessmarkednessandcorrelation**].In the context of kickflip detection, it represents the proportion of true kickflips that are identified correctly.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

F1-score: This metric provides a balanced measure of performance by combining both precision and recall. It is calculated using the harmonic mean on both values and outputs a value ranging from zero to one, with values closer to one, indicating better performance.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

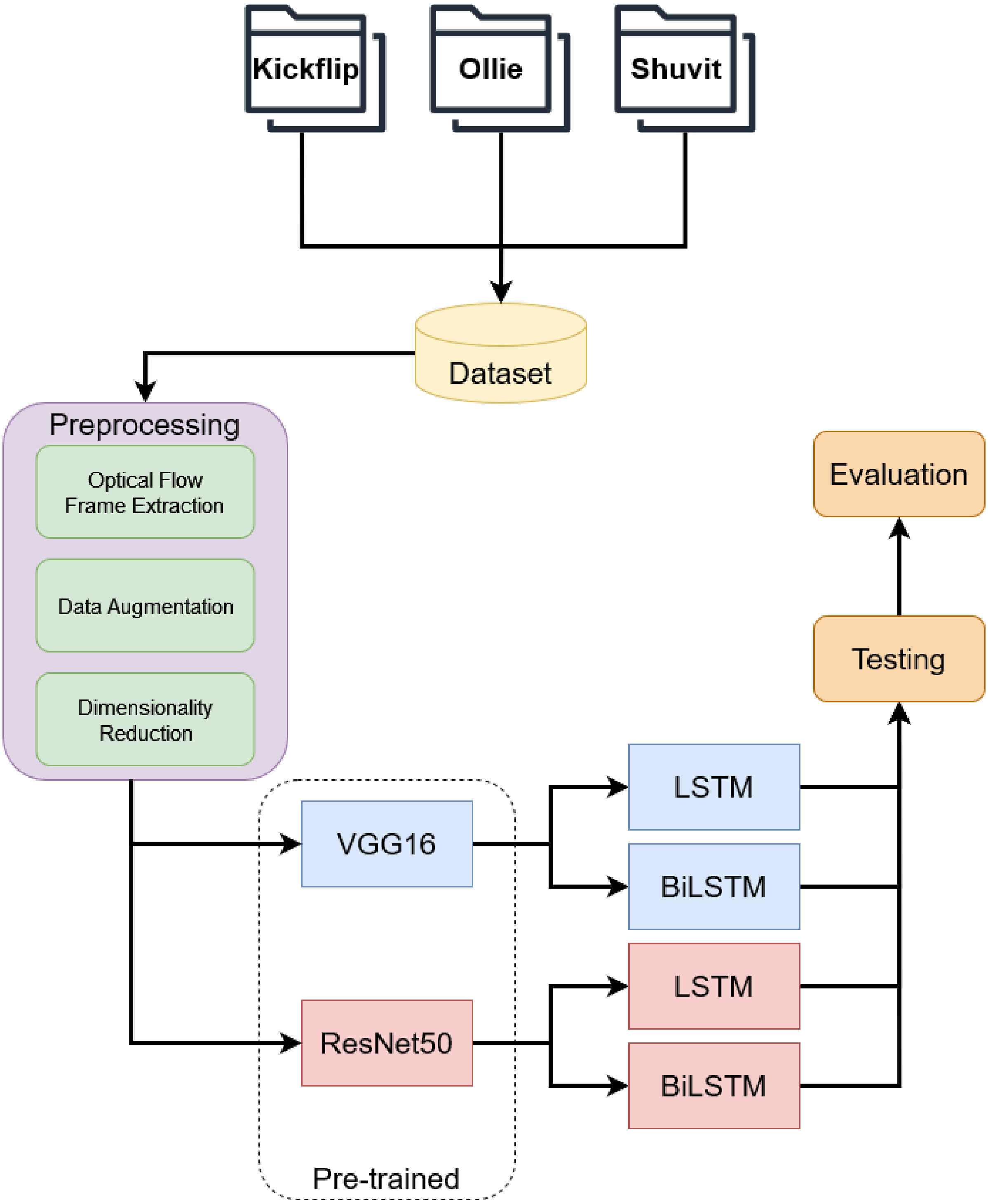


Figure 4.5 Artefact Architecture.

5 Implementation

5.1 Development Environment

Python: Python's large database of libraries, along with its wide use in Machine Learning, made it the ideal choice for developing this artefact. Furthermore, its large and active community made tackling problems and troubleshooting issues simpler. Prior experience using Python also contributed to this decision.

Tensorflow: Tensorflow is an open-source library, powerful in numerical computation and Machine Learning applications. It provides a rich toolset that allows for efficient development, training and deployment of Machine Learning models. Notably, Tensorflow comes with the Keras API, further simplifying the creation development by offering wide ranger of tools such as access pre-trained models, pre-defined layers and training and evaluation abstractions.

OpenCv (cv2): This open source library played a crucial role in the computer vision components of development, offering essential tools related to video processing, particularly during the frame extraction phase.

Matplotlib: This research made use of Matplotlib for its plotting and data visualisation functions, heavily used during the evaluation stage to properly visualise results or any other insights gained throughout.

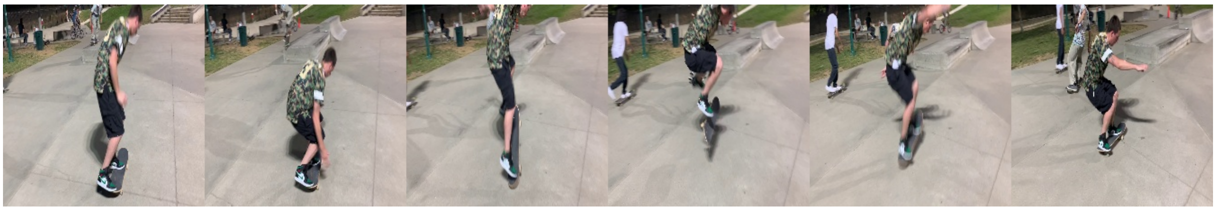
5.2 Dataset Split and Configuration

This study opted for a training-validation-testing split on the original dataset allocating 80% for training, 10% for validation and 10% for testing. The training set allowed the models to learn the patterns and relationships within the input data. The validation enabled a portion of the data to be used to evaluate the performance after every epoch. Finally, testing set provided a final assessment of the model's generalizability, by using data the model's have not seen before. The decision for this split stemmed from prioritising training data due to limited dataset size.

In consideration of Chen's (2023) [**SkateboardAIPaper**] research, which advocated for a sequence length of 45 frames per video, with each frame sized at 299x299 pixels, these parameters were initially evaluated. However, experimentation with such parameters revealed significant computational costs with negligible improvement in results. Consequently, this research struck a balance between costs and performance, by opting for a sequence length of 20 and a frame size of 224x224 pixels.

5.3 Frame Extraction using Optical Flow

The exploration of two frame extraction techniques aimed to find the most effective way to capture the crucial motions in a video through a series of frames. The optical flow method was hypothesised to capture frames with the most significant motion, unlike uniform sampling which often missed crucial parts of the skateboard trick, capturing more frames before or after the trick execution. An example of this behaviour at a smaller scale can be observed in Figure 5.1. Furthermore, experiments showed an increase in accuracy when using the optical flow method over uniform sampling, further supporting this hypothesis.



(a) Extracted frames using optical flow method.

(b) Extracted frames using uniform sampling.

Figure 5.1 Comparison of frame extraction between optical flow method and uniform sampling

This study employed Farneback's algorithm [**farneback2003two**] using the OpenCV (cv2) library, to estimate the optical flow magnitudes on each frame. The selection process involved reading pairs of consecutive frames from a video, resized for faster processing and converted to greyscale to minimise noise caused by colour variations. The cv2 function `cv2.calcOpticalFlowFarneback()` performed dense optical flow estimation between these frames, using parameters such as pyramid scale, levels, winsize, iterations that can be found in Appendix values.

The function `cv2.cartToPolar()` converted the Cartesian flow vectors returned by the optical flow function into polar coordinates, discarding the directional information, while retaining only the magnitudes. Finally, the code computed the average magnitude and appended it to a list. This process iterated through all frames to select those with the highest average to be extracted from the video.

5.3.1 Data Augmentation

This research explored a number of augmentation techniques using the `ImageDataGenerator` library provided by Tensorflow. This process involved defining a number of parameters that influenced the augmentation patterns applied to each frame, as illustrated in 5.3. In order to expand the number of training samples, this study created multiple copies of the original dataset, each augmented with its own set of unique parameters controlling image shifts, brightness, scaling and other transformations.

Figure 5.2 showcases an augmented trick sequence against its original, split into six frames for illustrative purposes. Experiments revealed that certain augmentation techniques such as image flipping and large rotational shifts disrupted the visualisation of the trick, negatively impacting model performance, thus these were removed or adjusted from the augmentation parameters.

(a) Augmented frames.

(b) Original frames.

Figure 5.2 Comparison of Augmented sequence against original

5.4 Feature Extraction and Preprocessing for Training

After successfully extracting the most significant frames from each video using optical flow, the next step involved preparing the data for further processing. The main algorithm consisted of an iterative process responsible for resizing and normalising all frames before input to the pre-trained CNNs for feature extraction. The `cv2.resize()` function resized all frames to 224x224 to satisfy the VGG and ResNet input image requirements, before performing min-max normalisation on each frame by diving the pixel intensities values by 255.

Once resized and normalised, the pre-trained CNNs extracted features from each frame. With the final classification layers removed, the extracted features of

```
{
    "augment_params_1" = {
        'width_shift_range': 0.1,
        'height_shift_range': 0.1,
        'shear_range': 0.1,
        'rotation_range': 10,
        'zoom_range': 0.2,
        'brightness_range': (0.7, 1.3),
        'channel_shift_range': 20.0,
        'fill_mode': 'nearest'
        }
}
```

Figure 5.3 Augmentation Parameters

a singular image using VGG returned a shape of $(1 \times 7 \times 7 \times 512)$, while ResNet50, resulted in a shape of $(1 \times 7 \times 7 \times 2048)$. In both cases, the outputs were flattened using the Tensorflow `flatten()` function to transform the extracted features into a one-dimensional vector suitable to be fed into the dense layers of the network. Thus, VGG generates a final feature shape of (20×25088) for an entire video, while ResNet50 generates (20×100352).

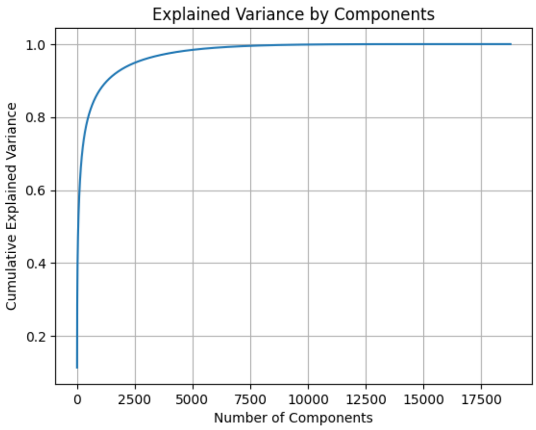
The NumPy function `np.save()` saved these extracted features along with their respective labels for training, validation, and test sets. This allowed for easy retrieval of the pre-processed data, eliminating the need to re-extract features with every new session.

5.4.1 PCA Dimensionality Reduction

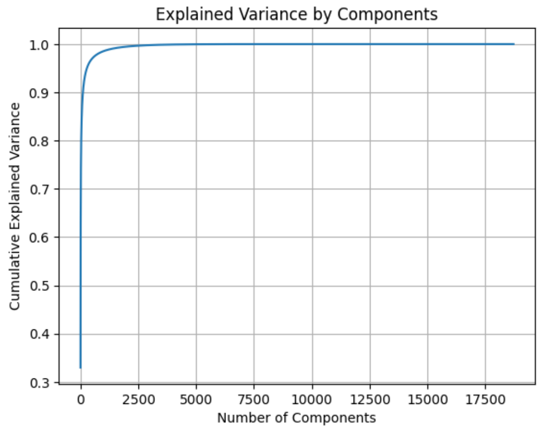
This research, leveraged Principle Component Analysis (PCA) dimensionality reduction, a technique, crucial in refining the feature sets produced by the pre-trained models. PCA a statistical technique that transforms features into a lower-dimensional space, where the components, known as principle components are orthogonal to each other. These principle components are reorganised in a way that the first few components attempt to portray maximum variance from the original data [**dimensionality_reduction**].

This research, aimed to achieve 90% to 95% variance retention when determining the optimal number of components for reduction. This involved plotting two explained variance plots: one for the output of pre-training with VGG16 and another for the output of ResNet50, as illustrated in Figure 5.4. These plots display the cumulative explained variance against the number of components, allowing approximate selection of the number of components by examining the 90% to 95% percentiles of the plots.

Upon examining both plots depicted in Figure5.4, 2514 components were chosen for VGG, while 278 were selected for ResNet50. Consequently, this results in final feature shapes of (20, 2514) and (20, 278) respectively.



(a) Caption for image 1.



(b) Caption for image 2.

Figure 5.4 Overall caption for both images.

The application of PCA dimensionality reduction significantly improved training efficiency. By transforming the high-dimensional feature space extracted by pre-trained models into a lower-dimensional one, PCA effectively reduced the computational costs during training.

5.5 Hyperparameter Optimisation

The architectures described in this study represent the best versions of themselves after a number of iterations and optimisations. To speed up the time-consuming task of hyperparameter tuning, this research leveraged the power of Optuna, a hyperparameter optimisation framework [**optunaFramework**]. Optuna leverages a Bayesian optimisation algorithm that runs through a number of iterations, observing the performance of past configurations and strategically selecting parameter combinations such as learning rate, dropout rate and neuron count.

This iterative approach enables Optuna to navigate the search space and converge on optimal hyperparameters for a specific model and dataset. This library was integrated within a custom-built development environment specifically designed to identify the optimal hyperparameter configurations for multiple model architectures and their corresponding input data.

5.6 Training History

Figure shows the training and validation loss curves for the models that achieved the best test accuracy within each architecture. Additionally, the vertical dashed lines indicate the epoch at which the early stopping callback restored the models weights, preventing overfitting.

All models employed the Stochastic Gradient Descent (SGD) optimiser throughout the training process due to excessive overfitting observed with Adam in Section 6.1.1. While SGD performed better than Adam in the context of classifying skateboard tricks, it required longer training times to converge.

5.6.1 Training Process and Hyperameter Tuning

Throughout the training phase, Optuna took the responsibility of initialising the model's hyperparameter configurations. Leveraging its Bayesian optimisation algorithm, Optuna provided initial configurations, which served as starting points for the iterative training process.

In this iterative process, each model's hyperparameters underwent continuous evaluation and refinement after every iteration at an attempt to improve accuracy. Performance metrics such as loss curves, confusion matrices and F-scores, were monitored closely to assess the effectiveness of each hyperparameter configuration.

Common pitfalls observed, included overfitting, indicated in model loss graphs when validation loss begins to rise, and fluctuations in performance metrics such as accuracy and F-scores. These fluctuations often signify an unstable learning process, suggesting the need for hyperparameters reconfiguration. Finally, with every iteration, the hyperparameters and performance scores were saved for future analysis and comparison, aiding in the refinement of upcoming iterations.

5.7 Callbacks

5.7.1 Early Stopping

An early stopping callback reduced overfitting and saved computational resources. This method monitored the validation loss at every epoch and halted training if improvement stopped after a predetermined patience value. All experiments investigated a patience of 10 epochs, based on the observation that the models were unlikely to improve after 10 epochs, with no validation loss advancements.

5.7.2 Model Checkpoint

This study incorporated model check pointing in the training process to save intermediate models after every epoch. This implementation was configured to monitor the validation loss and only save the model when it showed an improvement, allowing a seamless resumption of training in case of interruption.

5.8 Baseline Model Approach

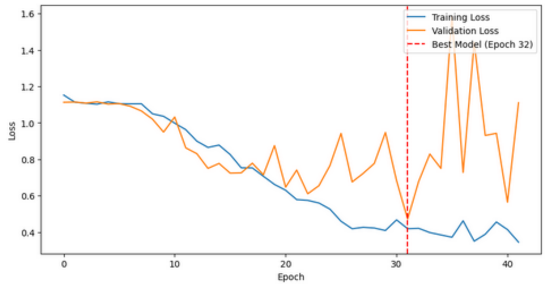
A baseline model approach defined during development of the artefact allowed comparisons to be made with each iteration of the models. The first baseline model

6 Evaluation

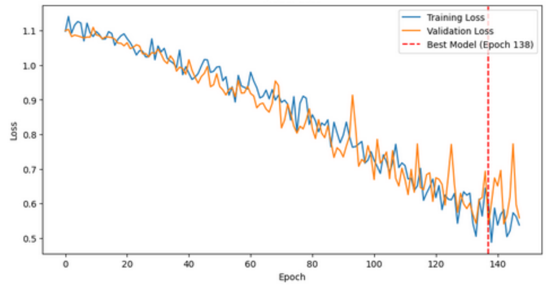
6.1 Experiments

6.1.1 Choice of Optimiser

This study compared the results of Stochastic Gradient Descent (SGD) and Adam [**adam_paper**] as optimisation algorithms during training. The results suggest that SGD may be more effective at reducing overfitting compared to Adam. This improvement is illustrated in Figure 6.1, using the VGG-BiLSTM architecture. While, Adam initially shows promise with good generalisation, it quickly encounters a pitfall, where the validation loss begins to climb, indicating overfitting. In contrast, the SGD-trained model exhibits a more stable validation loss, likely due to its algorithmic stability and its capability to achieve small generalisation errors as explained in the study by Hardt et al. (2016) [**SGD_stability_paper**]. The results observed with the VGG-BiLSTM aligns with the insights discussed by Hardt et al. suggesting that models trained using SGD are less vulnerable to overfitting.



(a) Training and Validation Loss using Adam



(b) Training and Validation Loss using SGD

Figure 6.1 Model loss graphs for (a) Adam and (b) SGD optimisers.

6.1.2 The Effect of Data Augmentation

This research, also compared the effects of Augmenting data before feeding it into the sequence models. Table.. showcases the results

6.2 Applicability In Real-Time applications

For real-time applications like analysing skateboard events, the speed at which data is processed is crucial. To evaluate the model's applicability in such applications, this study measured the time taken to classify a single video for each architecture.

The slow processing speeds outlined in Table is caused by the computational expensive video analysis pipeline. This pipeline involves various steps: frame extraction using optical flow, feature extraction with models like ResNet50 or VGG16, dimensionality reduction using PCA and finally classifying the video from the sequence model. While they are relatively slow, they are still applicable in real-time events

6.3 Discussion

6.3.1 Challenges

Trained models often encountered difficulty in differentiating between both the ollie and kickflip compared to the pop shuvit. This challenge likely arises to their shared visual characteristics. Both the ollie and the kickflip involve common elements such as the initial pop of the skatebaord and the absence of rotation on the y-axis, which distinguishes them to the pop shuvit. The confusion matrix shown in Figure 6.2 presents the ResNet-LSTM model exhibiting slightly better classification accuracy for pop shuvits compared to ollies and kickflips.

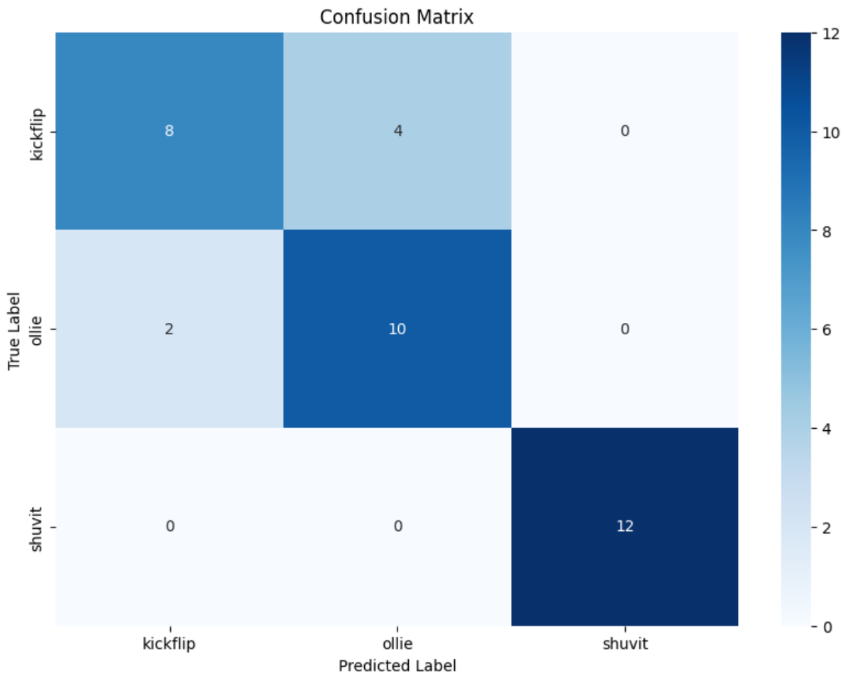


Figure 6.2 ResNet50-LSTM Confusion Matrix

7 Sample A

8 Sample B