UNIVERSITY OF OTTAWA

MASTERS THESIS

Efficient, Joint-Based Skeleton Action Recognition

Author:
Nicolas FLEECE

Supervisor:
Dr. Robert LAGANIÈRE

A thesis submitted in fulfillment of the requirements
for the degree of Master of Computer Science, Specialization in Applied AI

in the

VIVA Research Lab
School of Electrical Engineering and Computer Science

October 30, 2023

UNIVERSITY OF OTTAWA

Abstract

Faculty of Engineering

School of Electrical Engineering and Computer Science

Master of Computer Science, Specialization in Applied AI

Efficient, Joint-Based Skeleton Action Recognition

by Nicolas FLEECE

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
CIFAR	Canadian Institute For Advanced Research
CNN	Convolutional Neural Networks
EHPI	Encoded Human Pose Image
HMDB	Human Motion DataBase
I3D	Inflated 3D ConvNet
JHMDB	Joint-annotated Human Motion DataBase
LRCN	Long-term Recurrent Convolutional Networks
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
PA3D	Pose-Action 3D Machine
P-CNN	Pose-based Convolutional Neural Network
PoTion	Pose MoTion
UCF	University of Central Florida

Chapter 1

Introduction

People interact with their environment in unique and nuanced ways, and throughout our lives, humans have learned to identify and categorize the different actions that we perform.

1.1 Action Recognition

For humans, the problem of human action recognition is rather simple. We use past experiences throughout childhood and adult life to be able to pick out the various ways a person moves, and translate that into a familiar action that we have seen before. Combine that with objects that a person may be interacting with, and humans are remarkably good at discerning what actions other humans are involved in. However, as is with most things in the domain of computer vision, this ability does not translate well into the realm of artificial intelligence. The slightly different ways that people may perform these tasks add a layer of complexity that is difficult for a model to overcome.

1.2 Applications

Security is perhaps the most obvious example of action recognition usage. Security personnel are constantly on the lookout for suspicious individuals that may be of concern, or who are performing illegal actions. This can be as simple as trying to find those who are shoplifting in stores, where the CCTV footage can be used live to find those who are actively stealing from stores. It could also be something more complex, such as security checkpoints in airports, where screening officers are

constantly watching for suspicious individuals. In this case, a system capable of analyzing the way every person acts and pointing out those who it sees as suspicious could greatly assist security personnel.

Health care is a slightly different, but nonetheless very interesting application of action recognition. A very large part of how action recognition can help those in the healthcare field is use in monitoring those who need around the clock care, primarily the elderly. If an elderly person chooses to live at home, the action recognition model may allow healthcare workers to, at a distance, manage many people and focus their attention on those who have been flagged as in danger of injury. This can often be done by very lightweight models [1].

Video summarization is perhaps not directly related to action recognition, but rather the study of action recognition is a very useful part of video summarization. If you must summarize a video where the main subjects tend to be humans, a large part of figuring out what is going on in the video is figuring out what action the person is performing, for example, for a summary to be something such as '*The person is fishing.*', the model must have some understanding of what fishing is. Similarly, if the main subject of the video is not a person, it may still be useful to know what those in the background are doing, for example '*A dog is sitting on a bench, there are people doing yoga in the background*', the model again must have an idea of what yoga is, and how a person performs said actions.

1.2.1 Ethical Issues

As with most applications of artificial intelligence, computer vision cannot be researched and discussed without taking into account the ethical issues that surround it. With AI being such a quickly evolving space, it is crucial that any researchers be aware of these issues. In this section, I will focus particularly on how it may affect action recognition, touching on other areas of AI in general to further illustrate my points.

Privacy can become a concern in many areas, the healthcare example given previously in this chapter is one of the most obvious. With elderly people, often one of the draws to staying in their own private homes is the privacy that it offers; if the action recognition model is to be used to ensure that they remain safe, it must come

with some removal of privacy. There is also a question of what happens to the data of a person who is using this kind of service. Since the data is almost certainly sent to a server to be processed, what kind of data retention policies might this company have in place, are they sharing this data with others, or is the data going to be used to train future models. These are all issues that often follow AI since the training of models requires such a massive amount of data, and in action recognition this can contain people who are not necessarily aware of their data being used in such ways.

Bias is one of the most common ethical issues in AI. In artificial intelligence, and computing in general, the principle of "garbage in, garbage out", is commonly used to illustrate that if the inputs into a model are not of high quality, the outputs will not be of high quality. This can often be the case in datasets where say a group of people are not accurately represented, and while the model itself is not discriminatory, it will follow the data it has been given. For example, airport security has been scrutinized in the past for singling out individuals of particular races or who look a particular way. This may mean that if a model is being constructed that searches for people who may be flagged later in security, the majority of positive flags that were screened further would be of this group of people. The resulting dataset that is constructed would be biased against this group of people, therefore resulting in a similarly-biased model. This type of issue has been shown in many different areas, another example being speech recognition models used by voice assistants not being able to recognize particular accents as they were not represented in the original dataset. These kinds of reasons are why it is crucial for researchers to be aware of and study their datasets when it comes to human data to avoid these biases and ensure that their data is well balanced.

Transparency is a rather difficult, and often nearly impossible problem to solve with modern AI models. Given the fact that these models at minimum have millions of parameters that all contribute to the complex calculations towards the output, deciphering exactly how they work and make their decisions is difficult. These models are often depicted as black boxes, where the only context we are given is what is input into the model and what the model outputs, and nothing in between. In the cases of something such as an airport security checkpoint, the model may mark a person as acting suspicious in a line of passengers. The model is not able to specifically

express what made the passenger appear suspicious, and it may even be incorrect in its assumptions. This means that the officer who is reviewing the flags set by the model has to make a decision that leads to one of two possible scenarios:

1. The model is correct, but cannot communicate its exact reasoning with the officer, the officer does not see what the model sees and a potential threat is ignored

2. The model is incorrect, but the officer thinks that he sees something, and a person who is not a threat is put through unnecessary screening, and other potential threats may not be screened

1.3 Challenges

Human action recognition is a very difficult task that comes with many issues, some of which continue to be major challenges moving forward with very complex modern models.

Backgrounds are often not static in videos. Often they contain a lot of data that is ever changing and can contain other secondary subjects performing actions that may not be relevant to the subject that we are trying to determine the action of. While humans are very good at focusing on the person who is performing the action and ignoring things occurring in the background enough to not get confused, AI models do not have this inherent ability and often can get confused from background changes, and effectively must both identify the person and determine what action they are performing within the same model. This can be further worsened by any camera movement, resulting in both the subject moving throughout the frame as well as the background moving behind the subject.

1.4 Problem Definition

The problem of human action recognition is defined as taking a video of a person performing a particular action, and passing it through a model to determine the

specified action the person is performing. Pose-based action recognition is the problem of performing this detection primarily using the skeleton data of the people in the frame.

1.5 Contributions

The primary contribution that this thesis provides is a novel intermediate representation that can be used in tandem with a simple CNN for use in the problem of human action recognition. This representation is designed such that it is invariant to: the global position of the person, the distance between the camera and the person, and any background information. This representation only utilizes the skeleton data of the person, specifically the joint angles of a person and how they change from one frame to another.

1.6 Thesis Structure

This thesis will begin by exploring the research relevant to action recognition in chapter 2. Next, a novel representation is proposed in chapter 3, and the experiments and their results are detailed in chapter 4. Finally, the conclusions and recommendations for future work are presented in chapter 5.

Chapter 2

Literature Review

2.1 Image Classification

Image classification is the precursor problem to action recognition. Without the ability to classify individual images, the ability to classify videos, which functionally are a list of individual images, would never have been researched as nearly every technique for action recognition can be tracked from some form of image classification task.

An example of this task is the CIFAR dataset [2], where the goal is to simply classify relatively low resolution images into either 10 or 100 classes, depending on the version of the dataset. This dataset is very well explored, and results have high 90% accuracy values through very complex models and modern techniques. However, some simple models such as EfficientNet [3] are able to achieve this above 90% metric while providing a model that is able to be trained efficiently and evaluate images quickly. The popularization of image classification led to the popularization of CNN's which are also actively used in action recognition, and particularly popular in intermediate representation models that will be discussed in section 2.8.1.

2.2 Optical Flow

Optical Flow, not dissimilar to pose estimation described in section 2.7, is often taken for granted in models that utilize it, however it can be computed in several different ways. An example of an optical flow technique is the method utilized by the I3D model [4] which will be discussed later in this chapter. This technique is known as TV-L optical flow [5], and utilizes a formula based on the total variation (TV)

regularization and the robust L^1 norm in the data fidelity term. While the methods described in this paper and many others are rather complex, the idea that provides performance improvements is that optical flow represents the movement of a person from one frame to another and eliminating background information which does not move, this is further demonstrated in figure 2.1.

(a) First frame (b) Second frame (c) Optical flow field

FIGURE 2.1: An example of the optical flow field **(c)**, resultant from the first and second frames **(a)** and **(b)**. The goal being that the background is not in the optical flow field, only the movement of subjects in the frames are considered. This particular example is utilizing the TV-L optical flow [5].

2.3 Convolutional Neural Networks

Convolutional Neural Networks are a key part of both this thesis and the larger domain of computer vision. Using these models, computer vision took a large step forward in the domain of image processing, and is still being developed by modern models in order to extract further performance [6].

2.3.1 Structure

The classic structure of a convolutional neural network can be studied through one of the first recognizable architectures; LeNet-5 [7], as shown in figure 2.2. This architecture is broken into a few key layers that are common in classical CNN's:

- Convolutions
- Subsampling
- Flatten
- Fully Connected

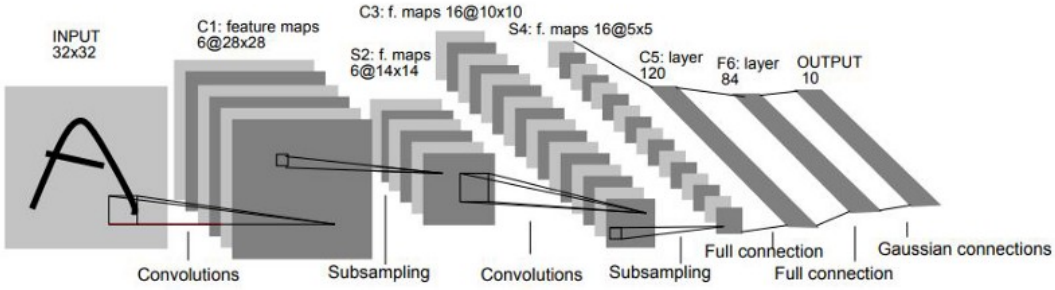


FIGURE 2.2: Classical LeNet-5 architecture [7] containing convolutional, subsampling, and fully connected layers.

The **convolutions** are the key part of how convolutional neural networks function, and they function using a **kernel**. This kernel is built as an NxN matrix of randomly initialized values, the matrix is then 'slid' across the entire image, and the values that the kernel 'sits on top' of are multiplied by the kernel values and then summed to produce an output cell in the output image. An example of a $4x4$ greyscale image and $2x2$ kernel is shown in figure 2.3. In addition to this process, sometimes the edges of images are 'padded' with zeroes in order to avoid the reduction in image size when a kernel reaches the edge of an image. This kernel can also be expanded to 3 dimensions in order to be utilized on video data rather than image data. This functionally works the same as 2-dimensional convolutions, sliding in three directions instead of two, however it results in much larger kernels that take up significantly larger portions of memory. This is because the back-propogation data must be then stored for 27 kernel data points rather than 9 (in the case of a $3x3$ convolution).

The **subsampling** (also commonly referred to as **pooling**) layers work in a similar way to convolutions, however instead of being trained, they are simply a standardized filter designed to down sample the feature maps exported by convolutional layers. This can be done through a few methods, average (as is used in LeNet) or max pooling are the most implementations. They simply take either the maximum or average value of the values covered by the subsampling kernel. The **flatten** layer simply transformes the outputs of the convolutional layers from shape $AxBxC$ to a one-dimensional vector of length $A * B * C$. This can then be passed through the **fully connected** layers, which are identical to those used in traditional fully-connected neural networks.

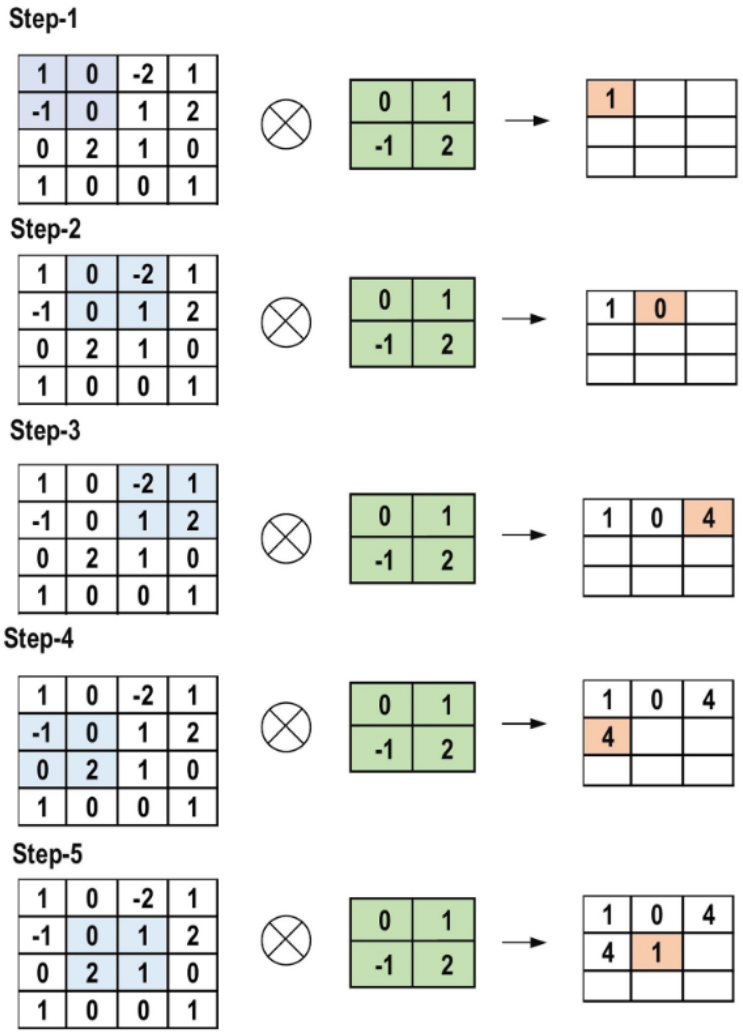


FIGURE 2.3: Example of kernel function as shown in Review of Deep Learning: Concepts, CNN architectures, challenges, applications, future directions [8].

2.3.2 Classic Architectures

AlexNet [9] is another common CNN architecture not dissimilar to the LeNet architecture [7] shown previously in this chapter. AlexNet provided two main contributions over the LeNet architecture. First, it provides a new method of training models on GPU, which involves splitting the parts of the model to be trained separately on different GPU's which can then be combined at the end (as shown in figure 2.4), which allowed more overall GPU memory to be utilized when training the model. Second, it provides a much deeper CNN architecture, where the LeNet architecture contains only 2 convolutional layers, the AlexNet architecture contains 5 total convolutional layers, three of which are stacked one after another after the second max pool layer. When it was introduced, AlexNet was the state-of-the-art CNN for image

classification and opened new doors with efficient multi-gpu trained models.

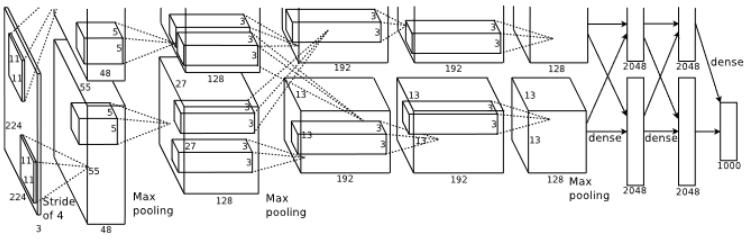


FIGURE 2.4: AlexNet [9] CNN architecture as it appears in the original paper [9]. The model is split into two separate top & bottom parts which are fed through individual GPU's and combined at the end.

VGG-16 [10] is another deep convolutional neural network model, again similar to those described previously in this section, however as AlexNet [9] built complexity on top of LeNet [7], VGG-16 seeks to do the same over AlexNet. The '16' part of VGG-16 refers to the number of convolutional layers, which is much more than the 5 present in AlexNet. While this added complexity is important, VGG-16 also focuses on keeping the individual parts simple, for each of their convolutional layers, the kernel is 3x3 with stride 1, and for each of the pooling layers, they utilize a 2x2 filter with stride 2. These convolutions are split into five different 'blocks' shown in figure 2.5, which are separated by these pooling layers, ending with the same structure of dense layers present in AlexNet.

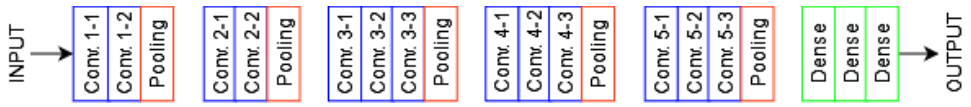


FIGURE 2.5: VGG-16 [10] architecture, with 16 convolutional layers split into 5 blocks, each block being separated by pooling layers, ending with three dense layers which provide output.

2.3.3 Modern Architectures

ResNet [11] deviates from this base CNN structure, aiming to improve on how these complex CNN's process data. With the evolution of CNN's naturally becoming more complex, the simplest approach was to add more convolutional layers (as can be seen from the transition from simple networks such as AlexNet [9] to more complex networks such as VGG-16 [10]). ResNet takes issue with this strategy, noting the problems of vanishing/exploding gradients becoming more prevalant and preventing the models from converging on proper solutions even if there is sufficient

complexity present. They aim to mitigate this issue via a simple "Residual block" shown in figure 2.6.

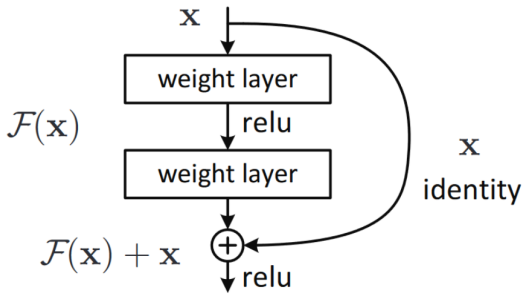


FIGURE 2.6: The Residual Block utilized by the ResNet [11] model. The input is added to the output of the convolutional block, resulting in an output that has both the output of the convolutions as well as the original input, which can then be fed into further blocks.

The function of the residual block is to make the model fundamentally easier to train. By separating the input into two parts $F(x)$ and x, the input to the next block will not have to rely on solely the outputs of the convolutional layer ($F(x)$), rather it can use the original input to the previous block x as well. This means that further convolutional blocks do not need to rely on the accuracy of $F(x)$, meaning that each block can focus on fitting individual patterns. This results in very complex networks that can still reliably find simple patterns, contrary to previous models where these simple details are lost to many stacked convolutions. This same method is employed in classic feed-forward neural networks as well using "shortcut connections", which functionally work in a very similar way to the residual blocks and has also proven to give performance increases.

Residual Attention Network for Image Classification [12] was another advanced technique to build off the existing CNN architecture that ResNet provided. This model aims to utilize attention modules, as shown in figure 2.7, which help the model to highlight points in the image that are more relevant to the image, and filter out background data. Again, building upon previous works, this model utilized the residual blocks used in ResNet [11], making it so that the model is able to be very deep and very complex without falling prone to vanishing/exploding gradients as easily.

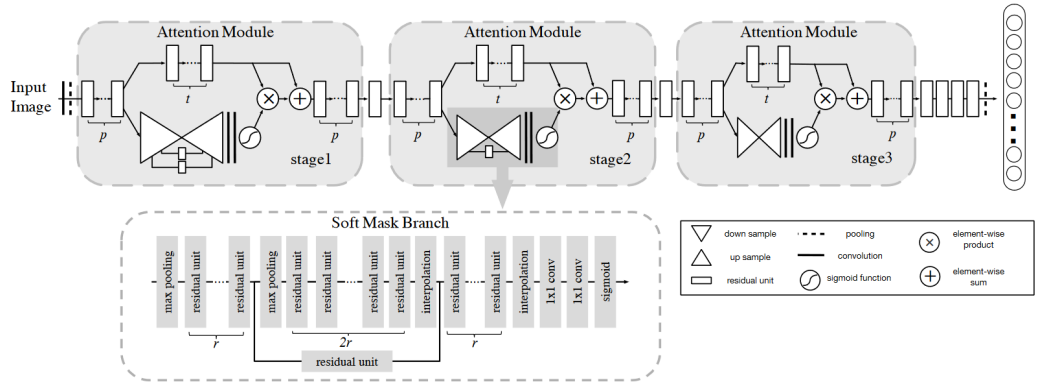


FIGURE 2.7: The model used in the **Residual Attention Network for Image Classification** [12], containing Attention Modules that are built from residual units as seen in the ResNet [11] model.

2.4 CNN Based Models

Naturally, the first approach to feeding video data into models is to process the raw RGB frames. This technique is derived from image classification tasks, where lightweight and relatively simple CNN's have been shown to have good performance when classifying single images. The logic would then follow that these types of models may be able to classify videos with fair accuracy, however they must be modified, which will be described further in this chapter.

2.4.1 CNN + LSTM Models

In very classic models, utilizing existing CNN architectures is a very simple process. The individual frames are passed through the CNN, producing a feature map for each frame. These feature maps are then flattened and passed into dense layers which produces an output. While very simple and fast, this model completely ignores any temporal activity, meaning that the model cannot determine how a person moves throughout a video from one frame to another. This would make differentiating some reversible actions such as running forwards vs running backwards very difficult.

Figure 2.8 shows the typical modern structure for this solution. After the features are extracted from each of the 2 dimensional CNN, they are passed through a LSTM. The goal of this LSTM module is to carry features from one frame to another and add some temporal element.

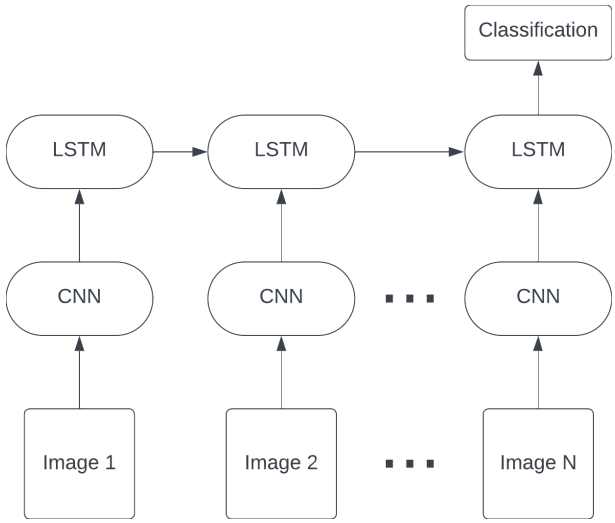


FIGURE 2.8: An example structure of a simple CNN-LSTM based model, each individual frame being individually fed into the CNN, and then passed to a LSTM.

The advantages of this model are that is is very lightweight and all of the individual parts are already well studied and efficient. This also means that the models are very lightweight, and relatively simple in comparison to more complex techniques.

The disadvantages of the model are also rooted in it's simplicity. The result of processing each image independently means that the interactions between frames is not very well represented. While the model is able to represent individual frame features very well, due to the fact that the feature maps are passed through the LSTM, classes that require specific movement from one frame to another are difficult to learn using this structure. Constructing these individual feature maps can also fall victim to background interference, meaning that a movement in the camera, or change in background could impact in a way that detracts from the main subject of the action more with this model than the other approaches discussed later in this chapter.

Long-term Recurrent Convolutional Networks [13], is a model constructed using this methodology. In the paper, they use the notation that each frame, x_i, is fed into the CNN in order to construct a fixed-length feature representation, $\phi_v(x_i)$. This is then passed into the recurrent sequence learning model. This is where the model differs from the previous example provided. In the LRCN model, the LSTM outputs at each frame are averaged to get the output class, rather than taking the last output. This removes any bias the model may have towards the later frames in long

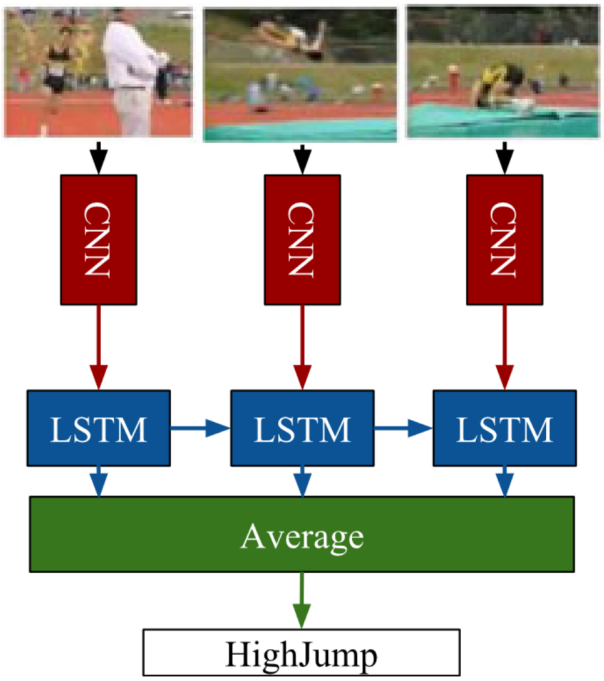


FIGURE 2.9: Action recognition structure for the LRCN model. [13]

videos. In addition to RGB frames, this model uses the optical flow feature, which easily adapts to this structure, replacing the RGB frames in figure 2.9. The LSTM structure is taken from **Learning to Execute** [14], which is a structure devised from the original LSTM model. The CNN, represented in the paper as ϕ, is described as a hybrid of the CaffeNet [15] (a variant of the AlexNet [9] model discussed in section 2.3.2) and the Zeiler and Fergus [16] models, which has been pre-trained on a large dataset.

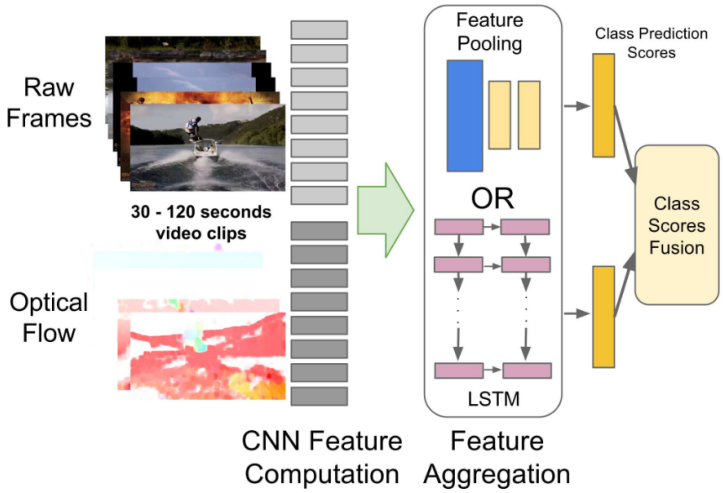


FIGURE 2.10: Overview of the Beyond Short Snippets: Deep Networks for Video Classification model [17]

Beyond Short Snippets: Deep Networks for Video Classification [17], is another approach to this structure, which explores a more complex deep-LSTM based module, as well as more classical feature pooling. Similarly to the previously discussed model, Long-term Recurrent Convolutional Networks [13], the model utilizes a combination of two popular CNN models, AlexNet [9] and GoogLeNet [18]. The paper did explore many more classical feature pooling architectures, and were proven to have good results, however these techniques were outperformed by the deep LSTM model. The paper utilized a deep LSTM architecture for the feature aggregation step, shown in figure 2.11, which further adds to it's complexity, moving it above the CNN-LSTM architectures described previously. In this deep-LSTM module, the outputs of each frame are passed into a LSTM module as in the previous model, but the ouputs are then passed up through 4 more stacked layers of LSTM's, after reaching softmax layers which are averaged to get an output. These 4 additional layers of LSTM's mean it is more able to infer data moving from one frame to another. This model additionally explored the uses of optical flow and found that it adds a great deal to the accuracy of the model.

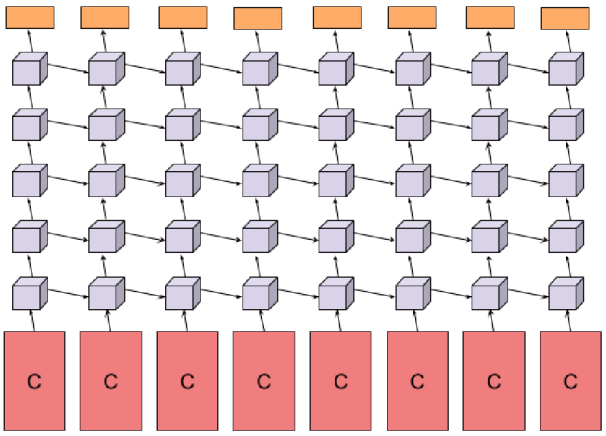


FIGURE 2.11: Deep LSTM architecture utilized by [17] in the feature aggregation step as shown in figure 2.10.

2.4.2 3D CNN Models

When considering how to handle videos without the LSTM component, the one of the first approaches that was developed is to utilize 3 dimensional CNN kernels. The function of these kernels when it relates to action recognition is that they allow for the model to easily encode local temporal data using the third kernel dimension.

The primary issue with these models is that they contain many more parameters over the 2 dimensional CNN models, meaning that they take longer to train and require more computing power as compared to the lightweight counterparts.

3D Convolutional Neural Networks for Human Action Recognition [19] was one of the original papers that proposed this model for the purposes of action recognition, and the greater topic of 3 dimensional convolutions. The general architecture of the model is shown in figure 2.12, and is extremely similar to that of 2 dimensional CNNs, with convolutional layers which are followed by subsampling layers.

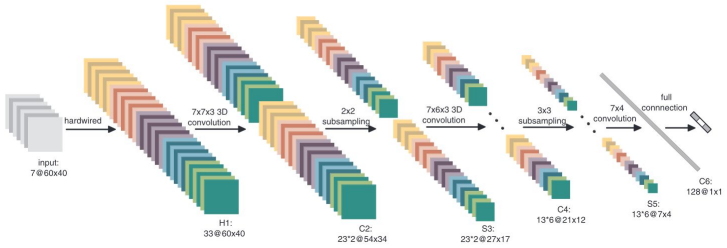


FIGURE 2.12: The original 3D-CNN action recognition model architecture proposed by **3D Convolutional Neural Networks for Human Action Recognition** [19], containing 3 convolutional layers, two subsampling layers, and one fully connected layer

The primary difference with this original architecture compared to 2D CNN's as we know them today is that it used rather large $7x7x3$ convolutions, as compared to the typical $3x3$ convolutions used in classical 2D CNN's. **Learning Spatiotemporal Features with 3D Convolutional Networks** [20] is a slightly more modern architecture that was proposed. They explore in great detail the effects of these sizes of convolutions and find that this size of convolutions are more effective than the previous methods and sizes.

Two-Stream Inflated 3D ConvNets, commonly referred to as I3D [4], is a modern variation on 3D CNN based networks. Similar to the previously discussed model [19], this model explores the viability of taking techniques used in 2D CNN models and applying them to 3D. However it takes a much more direct approach, stating that they take the square filters of size NxN and convert them simply to 3D filters with dimensions $NxNxN$, a process they describe as *inflating* the convolutions. This inflation of convolutions allows for I3D to replicate successful 2D CNN's in their structure and apply them to video with little modifications.

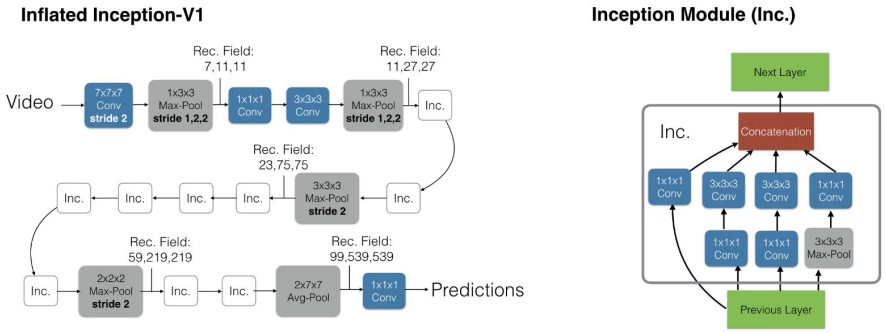


FIGURE 2.13: The model architecture used in the I3D model [4], where the Inflated Inception-V1 architecture (left) and it's detailed submodule (right) are shown.

2.5 Model Evolution

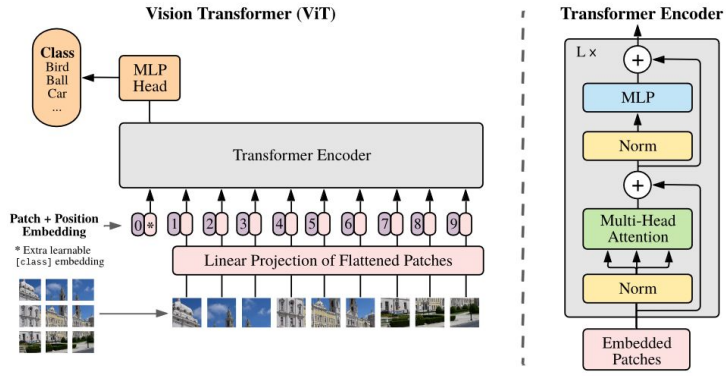


FIGURE 2.14: The original transformer model proposed in **An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale** [21], the image is split into fixed-size patches, linearly embed them, and add positional embeddings. It is then fed into a standard Transformer Encoder architecture.

Transformers for Image Recognition at Scale [21], extends beyond CNN's to explore transformer networks. While transformer networks are very easily applied to natural language processing tasks, it is not as easily applied to video and in particular action recognition. As depicted in figure 2.14, the model utilizes a standard transformer architecture in order to learn features that are useful for action recognition. The goal of this being to leverage previously well studied NLP studies that indicate the attention features of transformers are useful for focusing on the relevant data. Figure 2.15 shows this effect, the goal of this being to mitigate the challenges of handling background data interference as previously described in section 1.3. This

logic is then further expanded upon in many future models to extend this functionality, such as **Multiview Transformers for Video Recognition** [22] which explores using multiple separate encoders to explore multiple views.

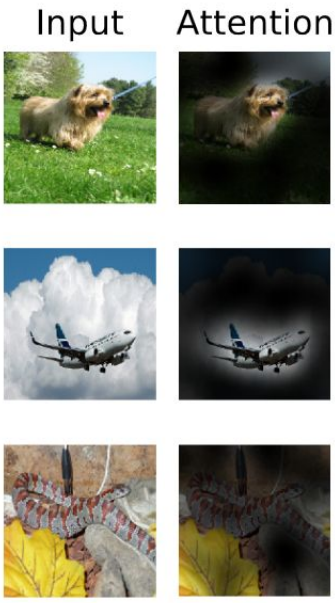


FIGURE 2.15: Example feature outputs of how a transformer utilizes attention to focus on the main subject of a video in order to greater identify actions as shown in [21]

2.6 Datasets

There are many action recognition datasets, some of which fit more specific use cases, and others which are more general and tailor to more in-the-wild data. For example the Charades dataset [23] focuses more on actions of people indoors and their interactions, whereas other datasets such as THUMOS [24] focus on many in-the-wild videos, where the location of the person can be different.

The Kinetics Human Action Video Dataset [25] is one of the primary in-the-wild action recognition datasets that are reported on by modern models. The primary advantage of this dataset over others that were published around the same time such as the UCF-101 dataset [26] is that as opposed to UCF's 101 classes and approximately 13,000 clips, the original kinetics dataset contained 400 classes and approximately 300,000 total clips which is magnitudes greater than other datasets of this type. This dataset has also been updated in 2020 to include 700 classes and

over 600,000 clips. The extremely large amount of these clips, as well as containing all of; singular person actions (eg. headbanging, stretching), person-person actions (eg. handshake, tickling), and person-object actions (eg. riding a bike) among others creates a dataset that is difficult for a model to determine what action is being performed.

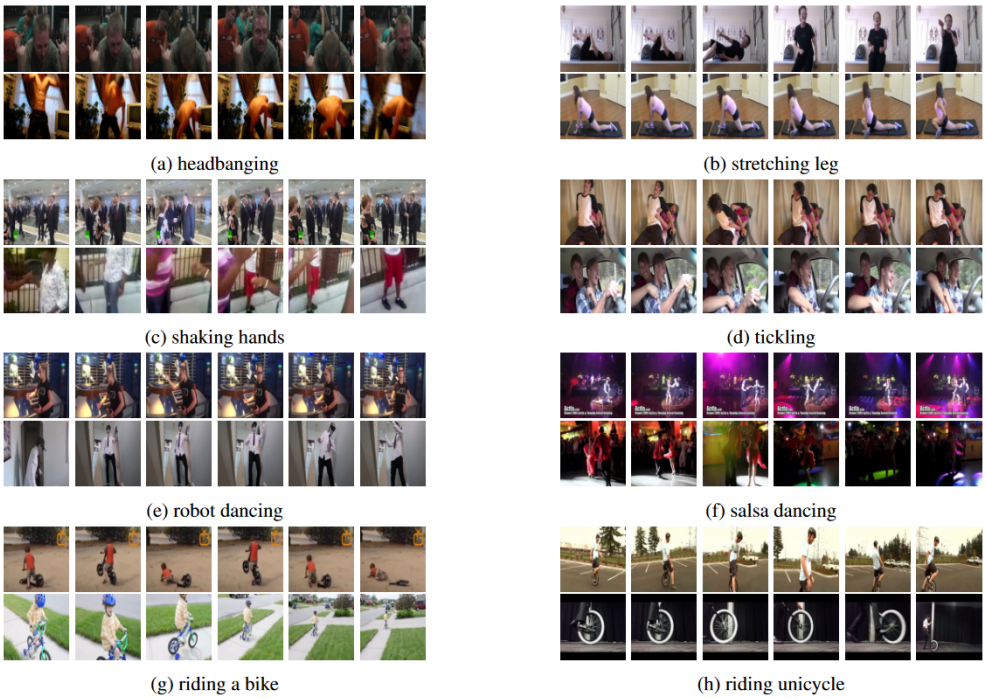


FIGURE 2.16: Example classes from the Kinetics dataset [25] which demonstrate the different singular person, person-person, and person-object interactions characterized in the dataset.

JHMDB [27] is the primary dataset that will be used in this thesis. The JHMDB paper does not actually propose an entirely new dataset, rather it proposes a subset of the HMDB dataset [28], with the addition of several features. While the dataset does offer more than only annotated poses, the main appeal of the dataset when considering the method used in this thesis is that they are fully annotated and adjusted poses to ensure that they are correct, meaning that the model can be independently be evaluated without having to consider the accuracy of the pose estimation model. Additionally, the dataset has been pruned such that the actions within the dataset only contain single person interactions, that lends itself very well to pose-based models, as generally only pose can be considered and the model can provide accurate results. Both of these features result in the dataset being highly popular

with 2D-pose based models, and in particular models that create intermediate representations with these poses, as the data that is extracted from pose is both accurate and relevant to the action.

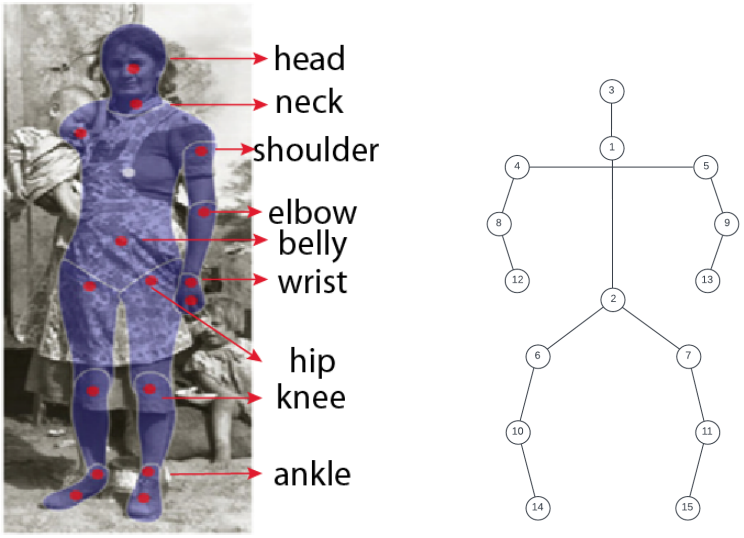


FIGURE 2.17: **Left:** JHMDB Annotated pose, each dot denoting a joint. **Right:** Simplified representation of this pose that will be used later in this thesis. Notably the foot joints are missing as they are not reliably labelled and are not used by the model. The indexes correspond to JHMDB joint indexes as if the subject were facing the camera.

2.7 Pose Detection

Throughout the majority of this thesis, and in some pose-based papers, the concept that there is some form of extracted pose from any given RGB frame is assumed to be accurate and complete. However the extracting of these pose features, especially in the wild, is a difficult task in and of itself. This means that when considering pose-based action recognition models (as will be done further in section 2.8) the accuracy of these techniques must be taken into account. Without an accurate pose model, it is impossible for these pose-based action recognition models to perform with any level of accuracy. It is also worth noting is that some training data can have manually annotated pose data, an example being the previously discussed JHMDB dataset [27], which utilizes manually verified pose data. A very common model utilized by pose-based models is **OpenPose** [29] (shown in figure 2.18) is capable of detecting the poses of many people within the frame. In addition to what is shown

in the figure, some models also utilize the joint heatmaps, which can also be easily generated by OpenPose. This is done through a modern technique using large CNN's and leveraging Part Affinity Fields, it is also allows for very fast real-time pose estimation.

FIGURE 2.18: Demonstrating the effectiveness of the OpenPose [29] model. The **top** image showing that it is capable of distinguishing individual people, the **bottom left** showing the Part Affinity Fields corresponding to the limb connecting the right elbow and wrist. The **bottom right** shows a zoomed in view of these Part Affinity Fields.

There are many pose estimation models, as it is in itself a problem in the domain of computer vision that is constantly evolving. These can range from transformer based models such as ViTPose [30] designed to maximize dataset metrics, to the very lightweight models such as MoveNet developed by TensorFlow [31], developed for the purposes of real-time pose detection through mobile devices.

2.8 Pose-Based Action Recognition

Pose-based action recognition models have been well studied, and have been one of the popular forms of action recognition model as people typically determine actions by examining how a person is moving. This is because by focusing on the pose (sometimes referred to as the skeleton) of the person, you are able to effectively mitigate the background effects that were discussed in section 1.3. This means that typically more lightweight models can be used as they are able to be pointed more

towards the main subject rather than filtering out background data. Of course this method also comes with challenges, notably that in testing in the wild, there must be effectively two models, one to extract the pose from the person(s) in the frame, and one to process this pose data and export an action. This can introduce another point of failure, but as discussed in section 2.7, 2D pose detection has been consistently improving to the point that high quality pose data from fast models is the norm.

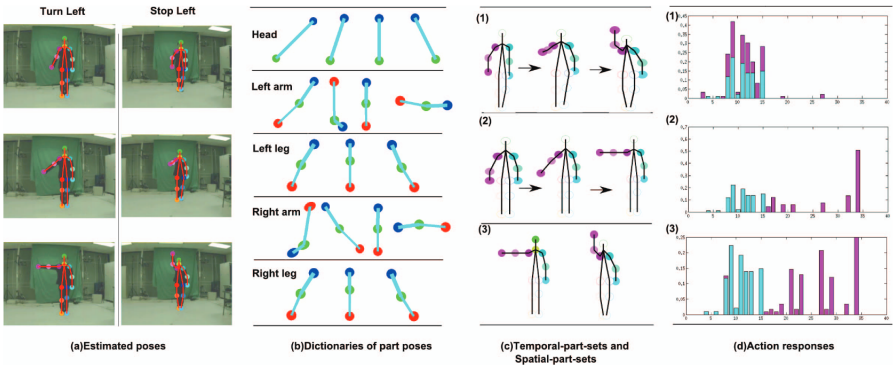


FIGURE 2.19: The overall framework of the action recognition model used by *An approach to pose-based action recognition* [32]. **(a)** & **(b)** show the esimated poses which are then used to create the dictionary of part poses. The temporal and spacial part sets in **(c)** are then represented in the histograms shown in **(d)**.

An approach to pose-based action recognition [32] was a paper that proposed a technique to pose-based action recognition that did not utilize any of the typical popular CNN's that are used in modern models. Instead this model utilizes a dictionary of part poses that is then used by a bag-of-words model, a model typically geared towards the domain of NLP. This representation is shown in figure 2.19, and while it performs well, it does suffer in that the bag-of-words models are not able to adapt to more sophisticated datasets.

Pose-Based CNN Features for Action Recognition (P-CNN) [33] is another model that utilizes the pose. Instead however, they use patches of the RGB frames centered on the various joints that have been detected, this is shown more in detail in figure 2.20. While this model does improve on typical models by using pose data, it does still struggle from the fact that it uses the raw RGB frame data, resulting in a model that is larger than would be desired.

Fusion-based architectures have had success in combining techniques used by basic 3D-CNN's, as seen in the I3D model [4], which as previously discussed in

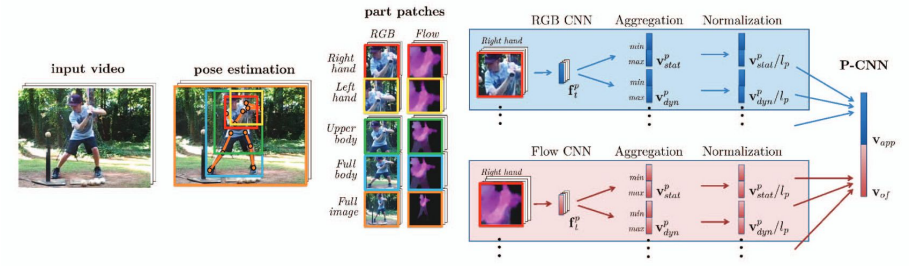


FIGURE 2.20: Illustration of P-CNN [33] feature construction. RGB & Optical Flow "Patches" are extracted around each joint, and sometimes containing multiple joints. These features are then passed through their respective CNN's, Aggregation, & Normalization and then concatenated to form the final P-CNN feature.

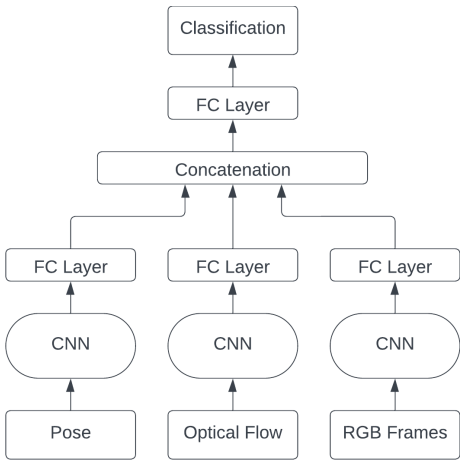


FIGURE 2.21: A typical fused architecture. Each of the Pose, Optical Flow, and RGB Frames are passed through individual 3D-CNN's, the outputs are then concatenated to achieve a final output.

section 2.4.2 utilizes the RGB Frames & Optical Flow in order to predict actions. Pose can be added to this architecture as shown in figure 2.21, where the pose data is added using some additional representation, the simple being the joint heatmaps exported from a previous model, and becoming more complex with intermediate representations that will be further discussed in section 2.8.1. **Chained Multi-stream Networks Exploiting Pose, Motion, and Appearance for Action Classification and Detection** [34] is an addition to this type of model, where instead of the classic fused architecture, the model has individual loss functions added to each of the outputs, increasing performance while not adding significant additional complexity to the model.

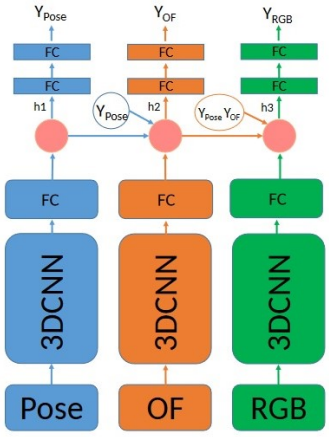


FIGURE 2.22: The chained architecture as shown in Chained Multi-stream Netowrks [34]. The model differentiates in that it has separate loss functions for each of Pose, Optical Flow, and RGB, which are chained together in a way that they can be individually optimized.

2.8.1 Intermediate Representations

Intermediate representations are the basis for what will be discussed in later sections of this thesis. Intermediate representations aim to reduce the pose data that has been extracted into simpler and more processable formats. This is generally done with the aim of using a more lightweight model (sometimes 2D CNN rather than 3D) that requires significantly less computing power. Of course, this kind of pre-processing comes with issues, notably that by converting the model into an intermediate representation, some data will inevitably be lost during the transition, so the problem definition shifts slightly to creating an intermediate representation that both allows for a lightweight model to be effectively trained on it and for the smallest amount of data to be lost in the transition.

Pose MoTion Representation for Action Recognition (PoTion) [35] proposed one of these intermediate representations, however they did it in a way that was unique in that they only considered the joint positions rather than the skeletons themselves. Namely, the model utilizes the joint positions of a person throughout each frame of video to construct 2 dimensional images that reflect the movement of each of these joints. This is done by stacking each joint heatmap onto one image, and colorize them according to the point in time the frame is extracted. This

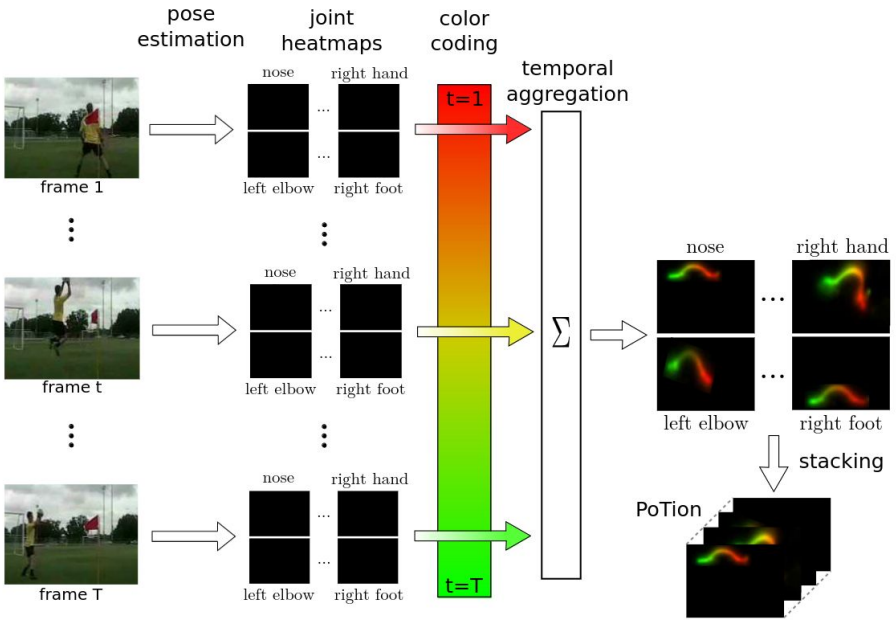


FIGURE 2.23: The illustration of the PoTion intermediate representation [35]. The input joint heatmaps are colored based on their time in the frame, and the frames are then concatenated to form the final movement of the joint throughout the video.

colorization technique is shown in more detail in figure 2.24. The overall representation construction is shown in figure 2.23, and shows how after the colourization is performed, the heatmap images are then stacked together. These stacked images can then be passed into a simple 2D CNN, which can be quickly and efficiently trained and performs rather well. In the paper they also explore adding this implementation as another input to the I3D [4] model in conjunction with the optical flow and rgb frames, which it showed to offer an increase in performance to the existing model. **Pose and Joint-Aware Action Recognition** [36] is a slight improvement on this model structure, they utilize a similar colorization scheme. Instead of feeding all of the joints into the model, they developed a joint-motion re-weighting network (denoted in the paper as JMRN), which allowed for the model to easily find the dependencies between joints. This joint selection procedure allowed for the model to offer improved performance over the original PoTion model.

Pose-Action 3D Machine for Video Recognition (PA3D) [37] takes a similar approach to the PoTion model, but flavours it slightly differently. Similar to PoTion model, it leverages joint heatmaps similar to that exported from the OpenPose pose detection model, however it does not colorize the joints and aim to insert them into

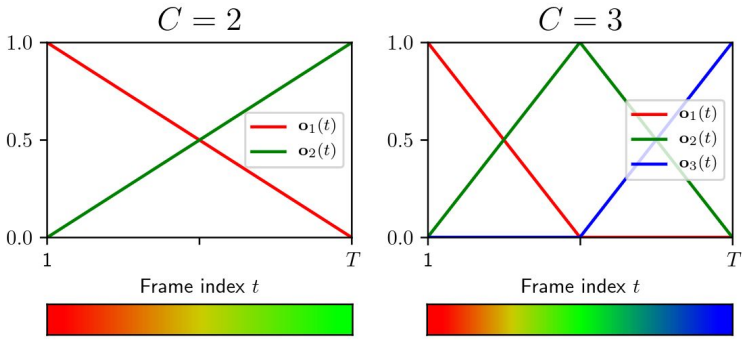


FIGURE 2.24: The colourization method utilized by the PoTion model [35]. As the frame index moves throughout the video, the colour of the joint shifts from one to another. This can be done for any amount of colours, denoted by C, the figure shows examples for C=2 and C=3, but the same logic holds for more than 3.

one image. A part of the model known as the *Temporal Pose Convolution*, shown in figure 2.25 is a core part of how the model functions. This is done through 1x1xN convolutions, which are run stacks of the joint heatmap images.

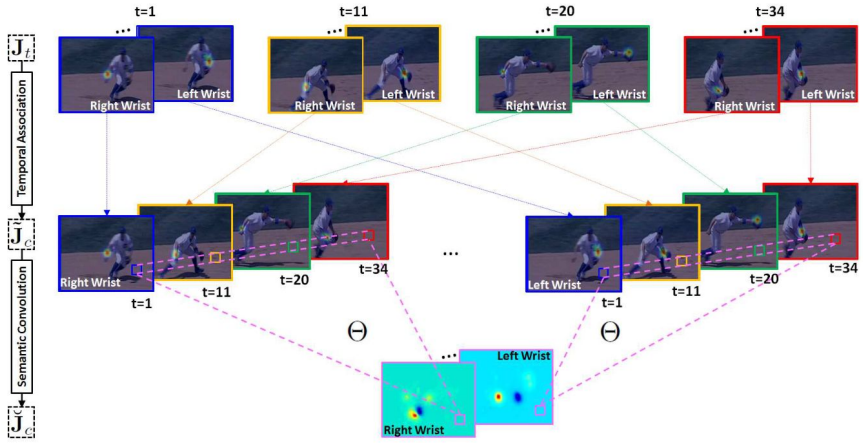


FIGURE 2.25: The main PA3D [37] model architecture, demonstrating the 1x1 convolutions used in order to construct the temporal cube.

Simple yet efficient real-time pose-based action recognition [38] was largely the inspiration for work done on this thesis. The goal of this paper was to provide a very lightweight and simple to understand intermediate representation that could be used by a very simple CNN to perform real-time action recognition. They do this by converting the skeletons into their unique *Encoded Human Pose Image (EHPI)*, which is a 2 dimensional grid where the x-axis is frame index, and the y axis is the joint, this is shown in figure 2.26. This EHPI representation can then be used with a very simple CNN to provide very fast and good results in order to process actions in

real time. There is one notable disadvantage in that it relies so heavily on the global positioning of the person in the frame. This means that the representation is very sensitive to things such as camera movement, where a slight movement results in the representation interpreting as the whole person sliding throughout the frame, however this could be mitigated via person detection to keep the person centered in the frame.

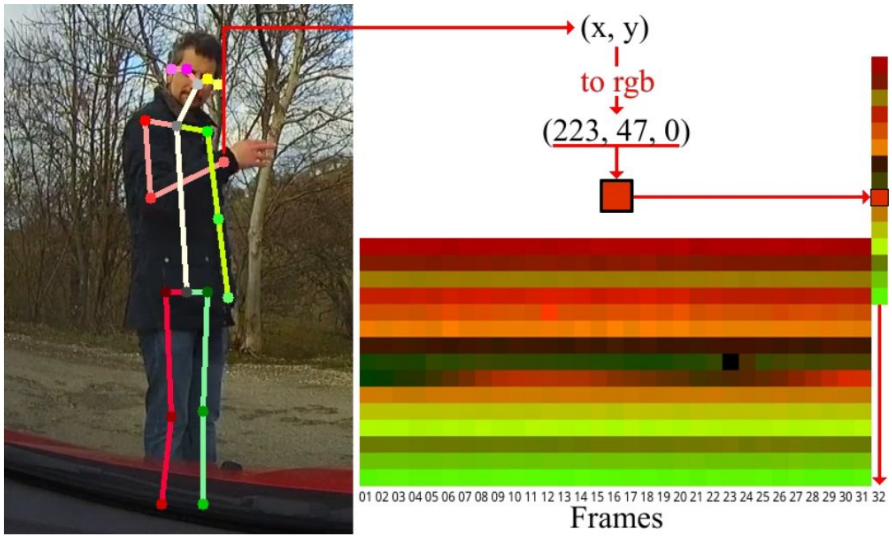


FIGURE 2.26: The EHPI representation used in the **Simple yet efficient real-time pose-based action recognition** [38]. The x, y coordinates of each joint are mapped to the red & green values of a pixel, all joints are then stacked to form a column of joint positions in a frame. Each frame is then placed next to each other to form the 2D representation.

Make Skeleton-based Action Recognition Model Smaller, Faster and Better [39] is yet another improvement on this intermediate representations, but with the particular focus on making the representations more resistant to both rotation & shifting of a person throughout the frame. As shown in figure 2.27, this is done through two features, the cartesian coordinate feature which was used in previous models in a similar way [38], and the JCD feature. The JCD feature is indifferent to shifting since all of the representation is aware of is the distance between any two given joints. This allows for easier generalization, however in the final model, both the cartesian coordinate and JCD features are used, as the authors determined that both were key in achieving high performance.

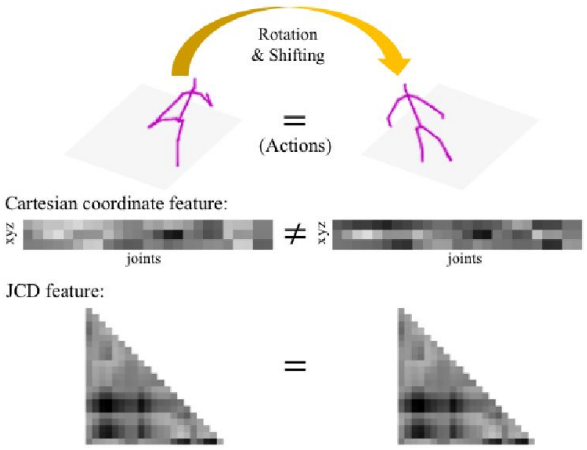


FIGURE 2.27: The representation used by the Smaller, Faster, Better model [39], this is split into two representations. The cartesian coordinates of each joint are encoded into a 2 dimensional representation, not dissimilar to previously discussed models [38]. The JCD feature is a similar representation, but instead of x, y, and z coordinates, uses the distance between two joints.

Chapter 3

Methodology

3.1 Dataset Selection

As has been stated previously in this thesis, the dataset that is used in the analysis of this model is JHMDB [27]. This dataset was selected for one primary reason, consistency. As was discussed in section 2.7, typical pose-based action recognition models utilize a two-step approach which involves first extracting pose data, then using that pose data in order to build intermediate representations. Usage of this dataset standardizes this first step since the pose data is included, rather than needing to be generated afterwards. This means that a more direct comparison can be drawn from other models since the input data is all the same. It is an understood fact throughout the remainder of this thesis that small gains may be able to be made through the addition of a more accurate pose model, however the scope of this thesis specifically focuses on the intermediate representation and corresponding models that process this representation.

3.2 Pose

The methodology throughout this section is dependent on pose information output from models that were described in section 2.7. These pose models have several outputs used by similar models such as joint heatmaps, however the main focus of this paper is the pure positions of the joints. These joints can be connected by lines to form "bones" which are then all connected together to form a "skeleton". This skeleton is shown in figure 3.1, where each of the joints are connected to form a human-like figure that can easily be created form the x & y coordinates of the joints.

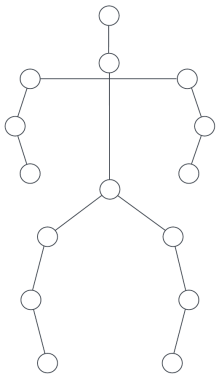


FIGURE 3.1: Example of how joints are connected through bones in a typical pose representations.

Using the JHMDB dataset, we simply take the existing pose implementation, the representation and indexes being shown in figure 2.17, with 15 total joints. Specifically in this thesis, we are concerned with bone-joint-bone connections, effectively representing the angle of the middle joint and how the bones around it move.

3.2.1 Joint Angles

The core of how the new representation will represent actions and movement has to do with the angle between two bones at any one given joint.

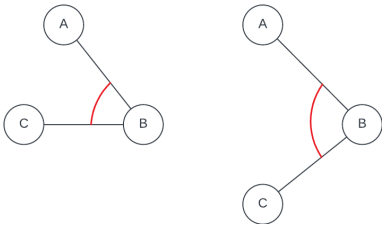


FIGURE 3.2: Two examples of three joints interconnected with two bones, and how the angle can change from one frame to another.

The first step in calculating the directional angle between the two vectors is to center one of the joints at the origin. Throughout this chapter, we will be referring to these two vectors as a and b. In figure 3.2, "a" would be denoting the vector $B \rightarrow A$ and "b" the vector $B \rightarrow C$. In this example, "B" will be the origin. This origin centering is simple and is done by the following equation 3.1, with each point representation as it was in figure 3.2 using the $B \rightarrow A$ bone as an example.

$$Vector A(x, y) = (A.x - B.x, A.y - B.y) \tag{3.1}$$

After processing both bones through this equation, it produces two vectors beginning at the origin. The goal then becomes calculating the angle between two of these vectors. This can be easily done by leveraging arctan. Calculating the angle of a given vector 'a' to the x-axis is simply done by using $\arctan(a.y/a.x)$. Once we have the angle of this vector to the x-axis, we can convert it to a positive value to ensure that the angle is being measured from a clockwise direction. Afterwards, we simply take the difference of the two vectors to determine the clockwise directional angle between two given vectors as denoted in equation 3.2.

$$\arctan(a.y/a.x) - \arctan(b.y/b.x) \tag{3.2}$$

The angle is calculated in a particular direction because if it were simply to be calculated agnostic of direction, it would be impossible to determine what angle a joint moves if it were to cross the 180° boundary. Table 3.1 demonstrates this issue, where a 180° rotation is observed from one right angle to another, but if non-directional angle is used, no change is represented.

a(x,y)	b(x,y)	Directional	Non-Directional
$(1,0)$	$(0,1)$	90°	90°
$(1,0)$	$(0,-1)$	270°	90°

TABLE 3.1: Example of how non-directional angle can result in incorrect rotation changes from one frame to another.

The resulting method means that some angles will be measured from the "outside" of the person and some will be measured from the "inside" as shown in figure 3.3. In practice, this does not have a large effect on the representation as the primary factor in this method is the change of an angle from one frame to another which is indifferent to how the angle is measured as long as the distance is consistent.

3.2.2 Joint Velocities

Once the joint angles for each frame have been extracted, the next step before implementation into our representation is to determine the change of a joint angle from one frame to another to determine the "velocity" of said joint angle. Generally this is simply done by calculating $angle(b) - angle(a)$, however there is one exception

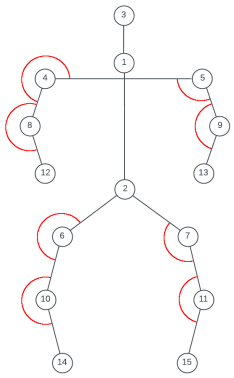


FIGURE 3.3: An example skeleton with JHMDB indices in each of the joints. In addition, angles have been shown as if they are being calculated in a clockwise direction, resulting in connections such as 5-4-8 having the angle on the "outside" of the person.

where the angle change crosses over the positive x-axis. This would be an example such as the difference between $a = 5$ and $b = 350$. The correct description would be that the angle moved $-15°$, however using that simple calculation would mean that the model would interpret this as a $345°$ change. This is most likely incorrect, as the more likely scenario is that the person moved $-15°$ rather than almost a complete rotation in the other direction. So we add logic such that if the difference between two angles is greater than $180°$ or lower than $-180°$, the most likely scenario is that it moved in the opposite direction for a shorter distance. So a change such as $270°$ would become $-90°$ and similarly a change of $-270°$ would become $90°$.

3.3 Novel Intermediate Representation

We can now construct the indermediate representation that will be used as input to our model. The model representation is very similar to that in the **Simple yet efficient real-time pose-based action recognition** [38] as described in section 2.8.1, which involves constructing a 2 dimensional image that is easily fed into a simple model. This is represented in figure 3.4, where one joint angle is taken from one frame of the video, and added to a column. This column consists of all of the data for one frame of the video, which is then added to the rest of the representation, eventually constructing the joint angles for each frame of the video.

As described previously in section 3.2.2, a key point of our representation is the joint velocities portion. These values can simply be "stacked" on top of the angles

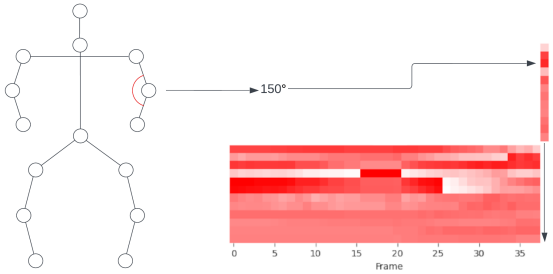


FIGURE 3.4: Example of how the intermediate representation is constructed. At each frame, the angles of each joint are taken and added to a column, each column is then stacked next to one another to form a 2 dimensional image.

to form the full intermediate representation. The logic is that for any given frame i, the column contains both the joint angles for the frame i, as well as their differences between frame i and $i + 1$.

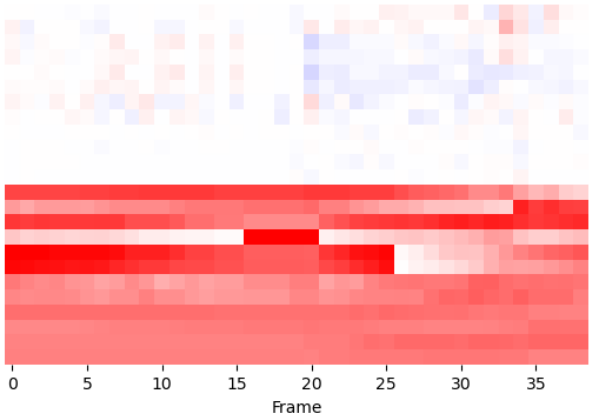


FIGURE 3.5: The finished intermediate representation, the top half contains the changes in joint angle from one frame to another (also referred to as "velocity"), and the bottom half contains the joint angles themselves. This is referred to as the "stacked" representation.

The specific joints that are represented in our intermediate representation are shown in table 3.2. These joints were chosen as they are major joints that form the typical "skeleton" visualization similar to that shown in figure 3.1 and 3.3.

3.3.1 Representation Features

The way the representation is constructed results in a representation that is:

- Rotation Invariant
- Scale Invariant

Joint A	Joint B	Joint C	JHMDB Indices
Face	Neck	Right Shoulder	$(3, 1, 4)$
Face	Neck	Left Shoulder	$(3, 1, 4)$
Right Elbow	Right Shoulder	Left Shoulder	$(8, 4, 5)$
Left Elbow	Left Shoulder	Right Shoulder	$(9, 5, 4)$
Right Elbow	Right Shoulder	Right Hip	$(8, 4, 6)$
Left Elbow	Left Shoulder	Left Hip	$(9, 5, 7)$
Right Wrist	Right Elbow	Right Shoulder	$(12, 8, 4)$
Left Wrist	Left Elbow	Left Shoulder	$(13, 9, 5)$
Right Shoulder	Right Hip	Right Knee	$(4, 6, 10)$
Left Shoulder	Left Hip	Left Knee	$(5, 7, 11)$
Right Hip	Right Knee	Right Ankle	$(6, 10, 14)$
Left Hip	Left Knee	Left Ankle	$(7, 11, 15)$

TABLE 3.2: All Joint connections used in our intermediate representations.

- Global Position Invariant

Rotation does not affect the representation since the angle of a joint from one frame to another only relies on the two bones, the two bones can rotate globally, what matters is only the angle between the two given bones. This effect is shown in figure 3.6, where the angle of interest does not change when another joint next to it is rotated, this can be generalized to any global rotation with the same logic.

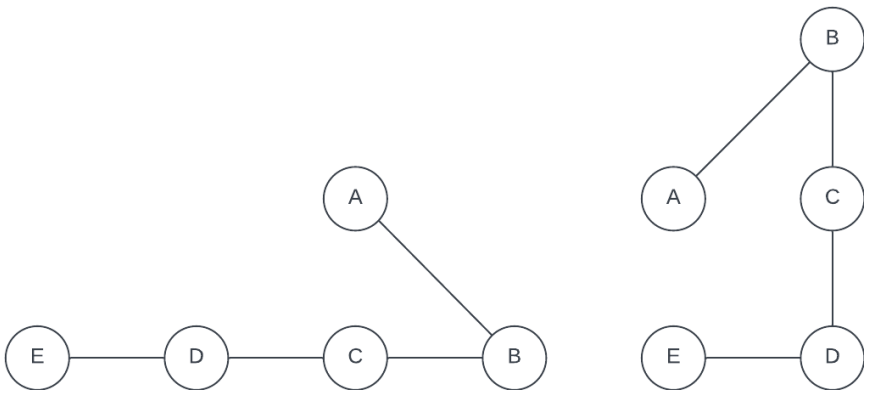


FIGURE 3.6: Example of the rotation invariance of our representation. The angle represented is $A - B - C$, and is unaffected by a rotation around $C - D - E$, remaining at 45°

.

Scale is avoided entirely as we do not take into account the length of the bones, rather only the angle of the joints. As was stated previously in this chapter, the bones are converted to vectors prior to computing the angle between them. This involves normalizing both bones to be length one, and this length data is not held for any

part of the representation. Therefore, the person can move closer or farther from the camera during the action, and it will have no effect on the representation.

Global Position is the final feature that the representation is invariant to. Similar to the previous features, the representation only cares about the angle of the specified joint, meaning that we can translate the person from one side of the frame to another with no change in the representation.

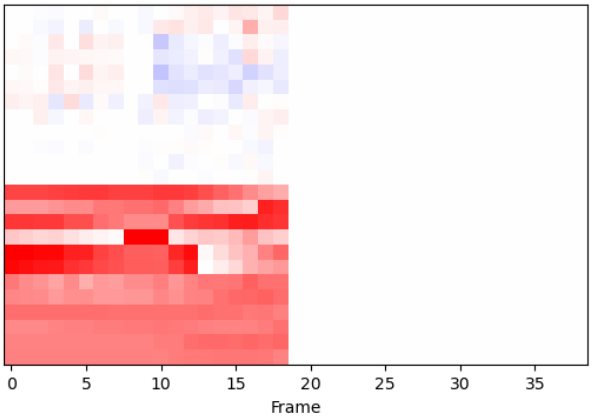
These three features of the model aim to mitigate background interference that has been previously discussed in the model. Specifically, they aim to reduce interference from movement of the camera. A person can move to either side, closer or farther away, and rotate the camera and there will be no effect on the representation.

3.3.2 Temporal Adjustments

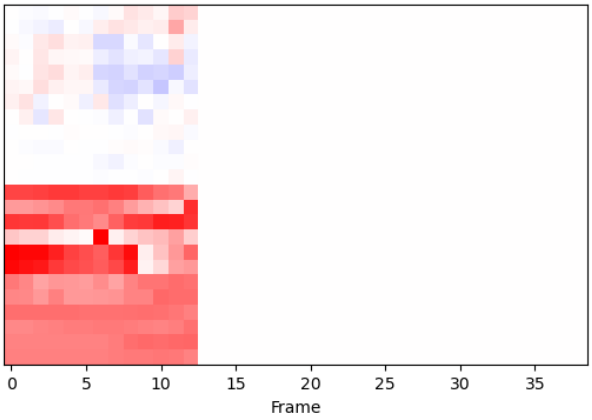
It is not unreasonable to assume that moving from only one frame to the next would result in some actions not being represented as well as other actions. Due to this issue, we add a temporal adjustment to our representation in order to better represent some of these actions. This is done by adding more channels to the data, with each of the channels "skipping" frames. This would mean that rather than having the angle difference representation moving from frames $1 \rightarrow 2 \rightarrow 3$, in the first additional channel it would move from frames $1 \rightarrow 3 \rightarrow 5$. The representation is then padded out to fit the same shape and added as another channel to the data. This is more clearly shown in figure 3.7 where the actions become more "compressed" and movements over longer periods of time become easier to visualize and process.

3.3.3 Model Architecture

The goal of this intermediate representation is to be able to use a simple 2-dimensional CNN which is able to run on simpler and less expensive hardware. The simplified architecture diagram in figure 3.8 shows the simple CNN architecture that is to be used. The model consists of 3 convolutional layer groups, followed by global average pooling and the final classification layer. Each individual layer and a more detailed breakdown is shown in table 3.3.



(a) N=1



(b) N=2

FIGURE 3.7: Two examples of temporal adjustment to the stacked representation. In both cases, the representation skips N frames in each step, meaning with a $N = 1$ the representation will ignore every other frame, $N = 2$ two frames, etc.

The model was trained using the SGD optimizer, with an initial learning rate of 0.01, momentum of 0.9, batch size 16, and weight decay 0.005. After each epoch, the learning rate was multiplied by 0.999, reducing the learning rate gradually over the training period.

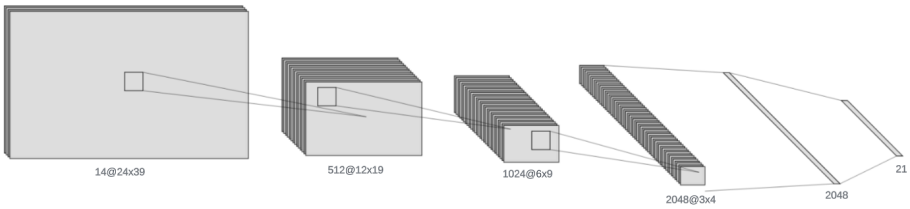


FIGURE 3.8: The basic model architecture used, very similar to that used in other models such as Potion [35], it is a simple 2 dimensional CNN that contains one fully connected layer at the end for classification.

2D Convolution @ 512 Channels 2D Batch Normalization ReLU 2D Convolution @ 512 Channels 2D Batch Normalization ReLU 2D Max Pooling
50% Dropout 2D Convolution @ 1024 Channels 2D Batch Normalization ReLU 2D Convolution @ 1024 Channels 2D Batch Normalization ReLU 2D Max Pooling
50% Dropout 2D Convolution @ 2048 Channels 2D Batch Normalization ReLU 2D Convolution @ 2048 Channels 2D Batch Normalization ReLU 2D Max Pooling
Global Average Pooling Flatten 21 Class Softmax Layer

TABLE 3.3: The detailed breakdown of the model architecture, split into 3 convolutional blocks and one dense layer used for classification. All convolutional layers utilize 3x3 convolutions, including a 1 pixel padding across all sides. The 2D max pooling utilized a 2x2 kernel, halving the size of the images after each convolutional block.

Chapter 4

Experimentation

4.1 Model Hyperparameters

4.1.1 Compute Resources

The model was mainly trained on a single **NVIDIA GeForce 3090** GPU, which is currently one of the highest end graphics cards. However, this GPU is significantly more compute power than is needed in order to train the model. The GPU used to train this model has 24 GB memory, whereas the model typically only takes 3.5 GB GPU memory at max. This would in theory mean the model could be trained on less modern, cheaper hardware such as the **NVIDIA GeForce GTX-1660**. The model is also almost certainly compact enough to be trained on purely CPU, however this would of course result in large training time increases, so for the purposes of this thesis it was not considered.

4.2 JHMDB Results

The dataset that the model will be evaluated on will be the JHMDB dataset. As previously stated in section 2.6, JHMDB is a good dataset to evaluate performance on pose-based models. The dataset contains 3 splits, each of which have a training and testing pair. Three independent models will be trained, one on each of the splits, without any previous pre-training, and with randomly initialized weights that will be seeded such that they're consistent from one run to another.

Table 4.1 contains the accuracy results of the previously discussed model and representation on the JHMDB dataset. As can be seen the model obtains on average

Split	Accuracy
1	58.209%
2	58.889%
3	58.113%
Average	**58.404%**

TABLE 4.1: Results on all 3 splits of the JHMDB dataset utilizing only our novel approach.

a 58% testing accuracy using only the novel approach presented. This is a sufficient accuracy to show that action recognition is possible with very lightweight representations that can then be processed via smaller models that can be run on less advanced hardware.

These accurate results do not tell the complete story however. As can be seen from the individual class F1 scores in figure 4.1, some classes the model is fairly good at predicting classes that have very clear movements associated with them such as *golf*, *pullup*, or *swing_baseball*. The significance of these classes is that the variance of the movement from one person to another is not very significant, so the model is able to learn the joint movements and apply that logic to other examples. Meaning the model is very good at recognizing these actions in any given environment.

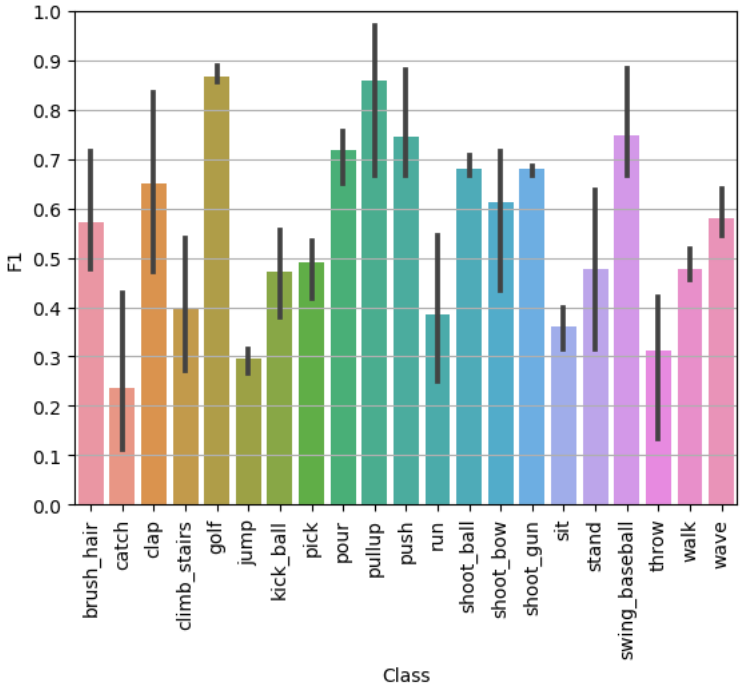
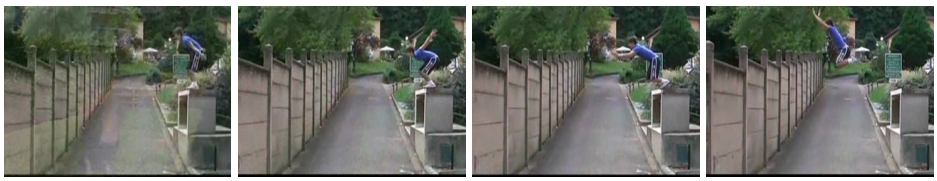


FIGURE 4.1: Detailed F1 score results on the JHMDB dataset, averaged over the 3 testing splits. Black bars show the corresponding maximum & minimum values attained in one of the splits.

The opposite factor is also something to consider. Figure 4.2 shows an example of three different sets of frames from the *jump* class. As can be seen from the frames, there are many different ways that a person may perform the *jump* action. A good example is the final two frames from figure 4.2(a), where the person simply moves throughout the global position of the frame, rather than actually moving any of their joints. Comparing to figures 4.2(b) or 4.2(c), where there are many joint movements throughout the frames, makes it difficult for the model to interpret them as the same and generalize to other examples of the same class. Figures 4.2(b) and 4.2(c) also demonstrate the variability in how these actions can be performed when they are general, where 4.2(b) shows more of a twisting motion, 4.2(c) is a more typical "straight on" jump.

(a)

(b)

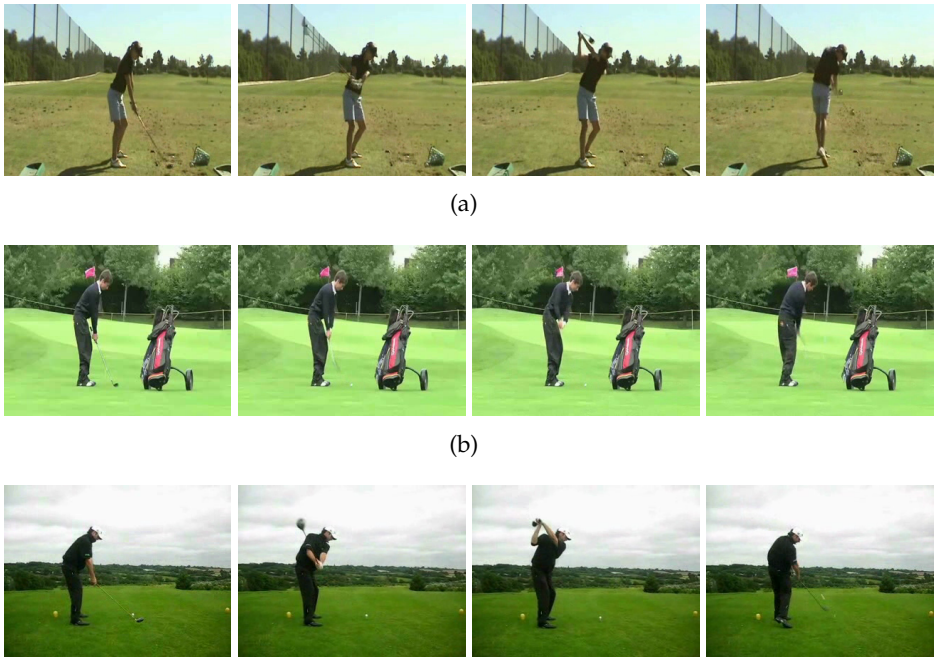


(c)

FIGURE 4.2: Three examples of the *Jump* class from the JHMDB dataset. The frames selected were four frames roughly evenly spaced out through the video.

This low variability in the *golf* class can also be demonstrated in a similar fashion to this. Figure 4.3 demonstrates this behaviour, where the action is nearly always performed in the same way where a person is standing, reaches back, twists and pulls the club forwards. Figures 4.3(a) and 4.3(c) are perhaps the best examples as they are swinging similar clubs, so the way that they move is almost identical, therefore meaning that the intermediate representation will also be similar. Figure 4.3(b)

is again a very similar movement, however it is a slightly different club meaning the movement is not as pronounced.

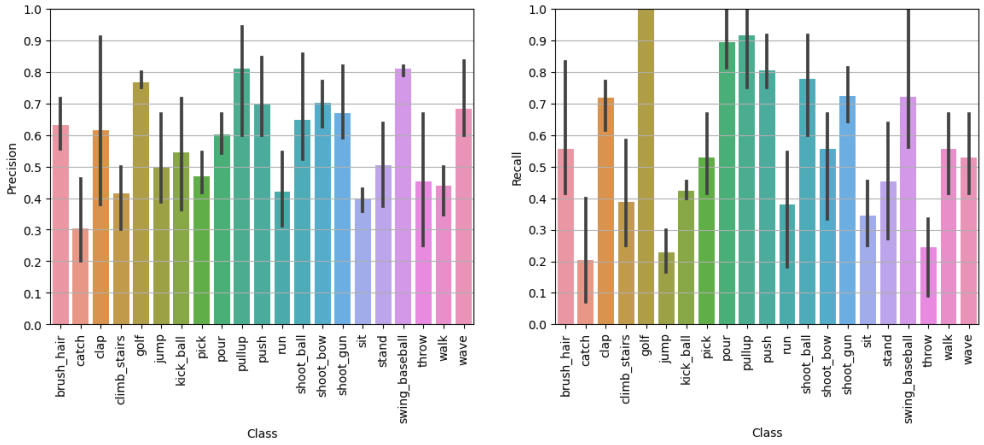


(a)

(b)

(c)

FIGURE 4.3: Three examples of the *Golf* class from the JHMDB dataset. The frames selected were four frames roughly evenly spaced out through the video.



(a) Precision (b) Recall

FIGURE 4.4: Detailed precision & recall results on the JHMDB dataset, averaged over the 3 testing splits. Black bars show the corresponding maximum & minimum values attained in one of the splits.

The detailed precision and recall values broken down by class are shown in figure 4.4, and in general reflect the same results as the F1 scores. The only notable

difference is the model tends to have higher recall differences and precision tends to be closer from one class to another.

4.3 Failure Cases

There are some specific failure cases that are worth observing.

4.4 Model Ablation Study

This section will present both different intermediate representation formats, as well as different model architectures. The natural first step to determining how effective the chosen architecture compared to others is to isolate each of the two halves (angle velocities & angles themselves). These results are presented in tables 4.2 and 4.3.

Split	Stacked	Only Angle Velocity
1	**58.209%**	48.888%
2	**58.889%**	44.444%
3	**58.113%**	41.132%
Average	**58.404%**	44.821%

TABLE 4.2: Comparison of results using only angle velocities vs the stacked representation.

The only angle velocity results show an average decrease in performance over all three splits of approximately 14% over the combination of the two. This is a very large decrease in accuracy, but is far from a negative result. What this result shows is that for some classes, the actual movement of the joint can be used in place of the joint angles and it is not necessarily required to have both in the representation. This is shown in figure 4.5, where we see that classes such as golf & pullup still have a relatively high F1-Score of 0.7 on average.

Split	Stacked	Only Angles
1	**58.209%**	56.343%
2	**58.889%**	**58.889%**
3	**58.113%**	56.604%
Average	**58.404%**	57.279%

TABLE 4.3: Comparison of results using only angles vs the stacked representation.

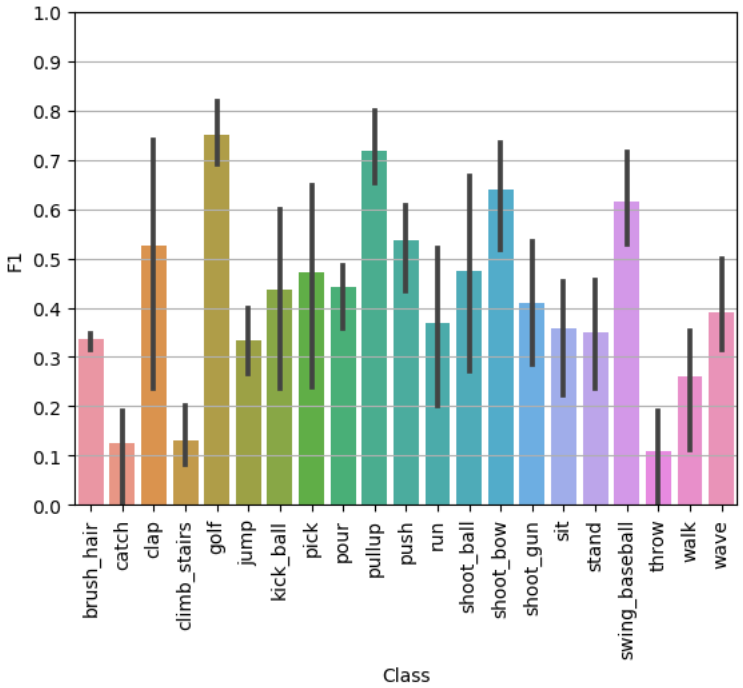


FIGURE 4.5: Detailed F1 score results on the JHMDB dataset using only the angle velocities from one frame to another.

Similarly, the representation that uses only angles and ignores the angle velocities shows a decrease in performance. However this is much less pronounced, and is only around 1%. This suggests that while the angle velocities do add value, it is not uniformly a large positive. However when considering how small the added data is in the stacked representation the small increase in accuracy is justified.

Another intermediate representation was also explored as is shown in figure 4.6, where rather than "stacking" the angles and velocities on top of one another, the rows alternate between the angles of corresponding joints and their velocities. The idea behind this representation being that the initial CNN filters would overlap between each individual joint angle and their corresponding velocity, meaning that the model may be able to learn these relationships a bit better.

The results of this alternating representation are shown in table 4.4. The results show on average a 4% decrease in performance when compared to the stacked model. This is almost certainly because the model benefits more from the convolutions being run on the angles and velocities independently and combining later in the model after pooling layers rather than on both the angles and velocities initially.

With this knowledge that the model typically performs better with the different

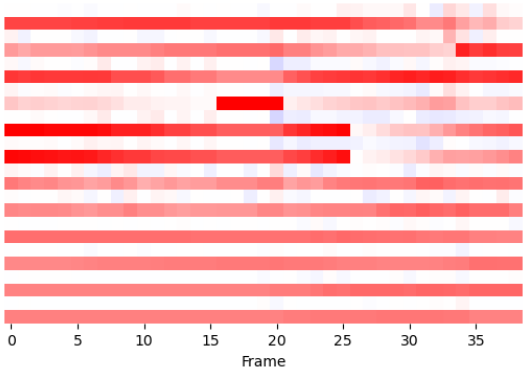


FIGURE 4.6: Alternative intermediate representation format where rows alternate between angles and velocities rather than one and then the other.

Split	Stacked	Alternating
1	**58.209%**	54.850%
2	**58.889%**	55.556%
3	**58.113%**	52.830%
Average	**58.404%**	54.412%

TABLE 4.4: Comparison of results using the alternating vs stacked representation.

features being stacked on top of each other and somewhat isolated, the next step was to see if further isolating the angles and velocities from one another would result in more performance gain. This resultant architecture is shown in figure 4.7, where we split the angles and velocities completely, mimicking the fusion-based architectures as were shown previously in figure 2.21, where these models tended to split RGB, Optical Flow & Pose data we similarly split our two pose joint features.

As can be seen from the results in table 4.5, once again this change to the model results in a decrease in performance by approximately 10%. This reduction in performance is most likely because of the increase in complexity to the model, it overfit the relatively small dataset much more easily, meaning that it was less easily able to generalize. It may also be due to the fact that in the stacked representation, the combination of angles and velocities can be "learned" as the convolutions combine them at each step, whereas the split model isolates them until the very end, potentially losing data in the process.

We can also examine utilizing deeper and shallower networks with this representation. A shallower model would allow us to further reduction in memory needed to

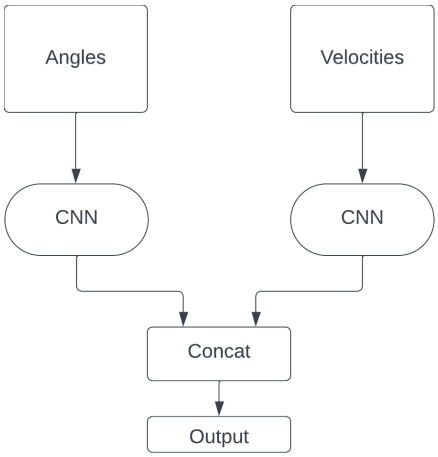


FIGURE 4.7: The split model architecture inspired by fusion models where the two different pose features are split and utilize independent CNN's.

Split	Stacked	Split Representation
1	**58.209%**	46.269%
2	**58.889%**	48.889%
3	**58.113%**	49.811%
Average	**58.404%**	48.323%

TABLE 4.5: Comparison of results using the split model and representation against the single CNN stacked representation.

both rain the model as well as a reduction in the hardware required to run the model in production scenarios. A larger model could potentially allow for an increase in performance while sacrificing some of the advantages described previously. Table 4.6 shows these results, the results from both shallow & deep models are outperformed by our proposed model. An exception is split 3 of the shallow model which outperforms our proposed model, however the average between the three splits is significantly lower. This is most likely due to the smaller model having slightly more volatility when it comes to each of the splits and results therefore varying more than our proposed model from one split to another.

4.5 Model Comparison

When comparing our model and representation to others that are similar, it is important to note the difference in expected performance of our model vs. complex state of the art models. For this purposes of this thesis, we compare our model to

Split	Standard Model	Shallow Model	Deep Model
1	**58.209%**	55.597%	55.970%
2	**58.889%**	57.407%	58.519%
3	58.113%	**59.245%**	53.585%
Average	**58.404%**	57.416%	56.025%

TABLE 4.6: Comparison of results using the final model compared to a shallower model using one less convoultional layer group, as well as a deeper model containing one additional convolutional layer group.

other pose-based models, not taking into account those complex models that take very large GPU's to train, as this would be an unfair comparison.

Model	Average JHMDB Accuracy
Chained [34]	56.8%
Potion [35]	57.0%
DynaMotion [41]	60.2%
EHPI [38]	60.5%
P-CNN [33]	61.1%
SIP-Net [42]	62.4%
STAR-Net [40]	64.3%
Pose & Joint Aware [36]	68.55%
PA3D* [37]	69.5%
Smaller, Faster, Better [39]	**77.2%**
Ours	58.404%

TABLE 4.7: Model comparisons that do not contain very complex models and utilize mainly pose data. * - PA3D uses additional data in it's representation as well.

As can be seen from the results in table 4.7, our model is not aiming to be the current state-of-the-art in it's category, however the performance is similar to the other models that approach the problem in a similar pose-based way. It is also notable, that our model is one of the only models that is both global position-invariant and scale-invariant, which is a large advantage of our model, and is by design as stated previously in this thesis.

Chapter 5

Conclusion & Future Work

In this thesis, we have provided a novel representation for use in action recognition. This novel representation is unique to many other models in that it is invariant to:

- Scale
- Global Position

We explore many different variations of this representation, concluding that the best representation contains both joint angle data and joint angle velocity data, however, only angle data also provides good results. It was also found that this data functions the best as a single image, with one data source stacked on top of another. This means that the combining of the angles and velocities can be learned by the convolutions slowly over the entire model, rather than typical fusion-like architectures that concatenate the different sources at the end.

With this representation, we were able to leverage a simple CNN model that is trainable on lightweight GPU's and is able to be tested on even lighter hardware. Despite the model being relatively lightweight, the model is able to provide 58.404% average accuracy over the 3 splits of the JHMDB dataset. This shows that despite the model not having been fed any data about how the person is moving globally, any data regarding the objects the person is interacting with, or any data about the environment it is able to fairly accurately predict what action the person is performing.

After further analyzing the results, the accuracy was shown to be slightly more nuanced than simply 58% accurate overall. The model was able to predict classes that had more consistent movements that varied less from person to person such as

golf or pullups, and tended to struggle with movements that can vary from person to person much more such as jump or catch.

5.0.1 Future Work

There is significant room for future work in this specific domain. Utilizing the skeleton data of a person to assist with action recognition is constantly evolving, and has continually been proved to help existing models improve on their results. However, a perhaps more interesting area of research that is not quite as popular is intermediate representations, which is where the "low hanging fruit" are. The first place of improvement would be to make the representation truly rotationally invariant, this would have to be done through 3-dimensional pose detection, and modifications of how changes in rotations are represented in the end image.

Another path forward is to more fully examine how real-time pose detection algorithms can be integrated with this simple model in order to obtain very fast and real-time results. This real-time computation is becoming more and more relevant in the world of smartphones, and examining how the model is able to be used on smartphones is crucial to determine the industrial applications. It is also worth noting that implementing this representation into existing complex models may be of interest in some applications, however the appeal of the representation being very lightweight and mobile applications not being as relevant cannot be ignored.

Bibliography

[1] H. Sun and Y. Chen, *Real-time elderly monitoring for senior safety by lightweight human action recognition*, 2022. arXiv: 2207.10519 [cs.CV].

[2] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[3] M. Tan and Q. V. Le, *Efficientnetv2: Smaller models and faster training*, 2021. arXiv: 2104.00298 [cs.CV].

[4] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[5] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," in *Pattern Recognition*, F. A. Hamprecht, C. Schnörr, and B. Jähne, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 214–223, ISBN: 978-3-540-74936-3.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] 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.

[8] L. Alzubaidi, J. Zhang, A. J. Humaidi, *et al.*, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25,

Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[12] F. Wang, M. Jiang, C. Qian, *et al.*, *Residual attention network for image classification*, 2017. arXiv: 1704.06904 [cs.CV].

[13] J. Donahue, L. A. Hendricks, M. Rohrbach, *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, 2017. DOI: 10.1109/TPAMI.2016.2599174.

[14] W. Zaremba and I. Sutskever, *Learning to execute*, 2015. arXiv: 1410.4615 [cs.NE].

[15] Z. Qin, F. Yu, C. Liu, and X. Chen, "How convolutional neural networks see the world — a survey of convolutional neural network visualization methods," *Mathematical Foundations of Computing*, vol. 1, pp. 149–180, Jan. 2018. DOI: 10.3934/mfc.2018008.

[16] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, 2013. arXiv: 1311.2901 [cs.CV].

[17] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4694–4702. DOI: 10.1109/CVPR.2015.7299101.

[18] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013. DOI: 10.1109/TPAMI.2012.59.

[20] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, *Learning spatiotemporal features with 3d convolutional networks*, 2015. arXiv: 1412.0767 [cs.CV].

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: 2010.11929 [cs.CV].

[22] S. Yan, X. Xiong, A. Arnab, *et al.*, "Multiview transformers for video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 3333–3343.

[23] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, *Hollywood in homes: Crowdsourcing data collection for activity understanding*, 2016. arXiv: 1604.01753 [cs.CV].

[24] A. Gorban, H. Idrees, Y.-G. Jiang, *et al.*, *THUMOS challenge: Action recognition with a large number of classes*, http://www.thumos.info/, 2015.

[25] W. Kay, J. Carreira, K. Simonyan, *et al.*, *The kinetics human action video dataset*, 2017. arXiv: 1705.06950 [cs.CV].

[26] K. Soomro, A. R. Zamir, and M. Shah, *Ucf101: A dataset of 101 human actions classes from videos in the wild*, 2012. arXiv: 1212.0402 [cs.CV].

[27] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, "Towards understanding action recognition," in *International Conf. on Computer Vision (ICCV)*, Dec. 2013, pp. 3192–3199.

[28] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: A large video database for human motion recognition," in *2011 International conference on computer vision*, IEEE, 2011, pp. 2556–2563.

[29] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[30] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, *Vitpose: Simple vision transformer baselines for human pose estimation*, 2022. arXiv: 2204.12484 [cs.CV].

[31] M. Abadi, A. Agarwal, P. Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[32] C. Wang, Y. Wang, and A. L. Yuille, "An approach to pose-based action recognition," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 915–922. DOI: 10.1109/CVPR.2013.123.

[33] G. Chéron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3218–3226. DOI: 10.1109/ICCV.2015.368.

[34] M. Zolfaghari, G. L. Oliveira, N. Sedaghat, and T. Brox, "Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[35] V. Choutas, P. Weinzaepfel, J. Revaud, and C. Schmid, "Potion: Pose motion representation for action recognition," Jun. 2018, pp. 7024–7033. DOI: 10.1109/CVPR.2018.00734.

[36] A. Shah, S. Mishra, A. Bansal, J.-C. Chen, R. Chellappa, and A. Shrivastava, "Pose and joint-aware action recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, pp. 3850–3860.

[37] A. Yan, Y. Wang, Z. Li, and Y. Qiao, "Pa3d: Pose-action 3d machine for video recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7914–7923. DOI: 10.1109/CVPR.2019.00811.

[38] D. Ludl, T. Gulde, and C. Curio, "Simple yet efficient real-time pose-based action recognition," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 581–588. DOI: 10.1109/ITSC.2019.8917128.

[39] F. Yang, Y. Wu, S. Sakti, and S. Nakamura, "Make skeleton-based action recognition model smaller, faster and better," in *Proceedings of the ACM Multimedia Asia*, ser. MMAsia '19, Beijing, China: Association for Computing Machinery, 2020, ISBN: 9781450368414. DOI: 10.1145/3338533.3366569. [Online]. Available: https://doi.org/10.1145/3338533.3366569.

[40] W. McNally, A. Wong, and J. McPhee, *Star-net: Action recognition using spatio-temporal activation reprojection*, 2019. arXiv: 1902.10024 [cs.CV].

[41] S. Asghari-Esfeden, M. Sznaier, and O. Camps, "Dynamic motion representation for human action recognition," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 546–555. DOI: 10.1109/WACV45572.2020.9093500.

[42] P. Weinzaepfel and G. Rogez, *Mimetics: Towards understanding human actions out of context*, 2021. arXiv: 1912.07249 [cs.CV].