**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

**SCHOOL OF SCIENCE**
**DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION**

**BSc THESIS**

# Human Action Recognition using Joint Trajectory Maps and Convolutional Neural Networks

**Simon J. Igiamou Perisanidis**

**Supervisors:** **Evaggelos Spyrou**, University of Thessaly Assistant Professor, National Center for scientific Research (NSCR)-“Demokritos”, Associate researcher.

**Panagiotis Stamatopoulos**, National Kapodistrian University of Athens (NKUA) Assistant Professor

**ATHENS**
**August 2020**

**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ**
**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

# Αναγνώριση Ανθρώπινων Ενεργειών με την χρήση Joint Trajectory Maps και Συνελκτικών Νευρωνικών Δικτύων

**Σίμων Τ. Ιγιάμου Περισανίδης**

**Επιβλέποντες:** **Ευάγγελος Σπύρου**, Επίκουρος Καθηγητής Πανεπιστήμιο Θεσσαλίας, Συνεργαζόμενος ερευνητής Εθνικό Κέντρο Έρευνας Φυσικών Επιστημών (ΕΚΕΦΕ)-«Δημόκριτος»

**Παναγιώτης Σταματόπουλος**, Επίκουρος Καθηγητής Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών (ΕΚΠΑ)

**ΑΘΗΝΑ**
**Αύγουστος 2020**

**BSc THESIS**

# Human Action Recognition using Joint Trajectory Maps and Convolutional Neural Networks

**Simon J. Igiamou Perisanidis**
**S.N.:** 1115201600051

**Supervisors:** **Evaggelos Spyrou**, University of Thessaly Assistant Professor, National Center for scientific Research (NSCR)-“Demokritos”, Associate researcher.

**Panagiotis Stamatopoulos**, National Kapodistrian University of Athens (NKUA) Assistant Professor

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

# Αναγνώριση ανθρώπινων ενεργειών με την χρήση Joint Trajectory Maps και συνεκτικών νευρωνικών δικτύων

**Σίμων Τ. Ιγιάμου Περισανίδης**
**Α.Μ.:** 1115201600051

**Επιβλέποντες:** **Ευάγγελος Σπύρου**, Επίκουρος Καθηγητής Πανεπιστήμιο Θεσσαλίας, Συνεργαζόμενος ερευνητής Εθνικό Κέντρο Έρευνας Φυσικών Επιστημών (ΕΚΕΦΕ)-«Δημόκριτος»

**Παναγιώτης Σταματόπουλος**, Επίκουρος Καθηγητής Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών (ΕΚΠΑ)

## ABSTRACT

Human action recognition is an important branch of human-centered research, which focuses on the automatic identification of human behaviours from images or video sequences. Motivated by the promising performance of Convolutional Neural Networks, and their applicability to the human action recognition task, in this thesis, we provide an effective method which is able to recognize human actions in a multitude of demanding scenarios.

Our approach is based on Joint Trajectory Maps and multi-stream Convolutional Neural Networks. The spatio-temporal information of the 3-D skeleton sequences are encoded into JTMs and fed into a deep three-stream ConvNet. Our architecture is trained and evaluated on the challenging PKU-MMD dataset and achieves competent results.

**SUBJECT AREA:** Computer Vision, Machine Learning, Human Action Recognition

**KEYWORDS:** Convolutional Neural Network, Recognition, Action, Skeleton, Joint Trajectory Maps

## ΠΕΡΙΛΗΨΗ

Η αναγνώριση ανθρώπινων ενεργειών είναι ένας σημαντικός κλάδος της έρευνας με επίκεντρο τον άνθρωπο, η οποία επικεντρώνεται στην αυτόματη αναγνώριση των ανθρώπινων συμπεριφορών από εικόνες ή βίντεο. Με κίνητρο την πολλά υποσχόμενη απόδοση των Συνελκτικών Νευρωνικών Δικτύων, και την εφαρμοσιμότητα τους στην αναγνώριση ανθρώπινων ενεργειών, σε αυτή τη εργασία, παρέχουμε μια αποτελεσματική μέθοδο που είναι σε θέση να αναγνωρίζει τις ανθρώπινες ενέργειες σε ένα πλήθος απαιτητικών σεναρίων.

Η προσέγγισή μας βασίζεται σε Joint Trajectory Maps και σε Συνελκτικά Νευρωνικά Δίκτυα πολλαπλών ροών. Οι χωροχρονικές πληροφορίες των τρισδιάστατων αλληλουχιών σκελετού κωδικοποιούνται σε JTMs και τροφοδοτούνται σε ένα βαθύ ConvNet τριών ροών. Η αρχιτεκτονική μας έχει εκπαιδευτεί και αξιολογηθεί στο απαιτητικό σύνολο δεδομένων PKU-MMD και επιτυγχάνει ικανοποιητικά αποτελέσματα.

**ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ:** Υπολογιστική Όραση, Μηχανική Μάθηση, Αναγνώριση Ανθρώπινων Ενεργειών

**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:** Συνελκτικά Νευρωνικά Δίκτυα, Αναγνώριση, Ενέργεια, Σκελετός, Joint Trajectory Maps

# ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον Επίκουρο Καθηγητή κο. Ευάγγελο Σπύρου που μου έδωσε την ευκαιρία να εκπονήσω υπό την επίβλεψή του, την πτυχιακή μου εργασία στο εργαστήριο Υπολογιστικής Ευφυΐας του Ινστιτούτου Πληροφορικής και Τηλεπικοινωνιών του ΕΚΕΦΕ «Δημόκριτος». Επίσης, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα του Πανεπιστημίου Θεσσαλίας κο. Ιωάννη Βερνικο για την πολύτιμη βοήθεια του στην εκπόνηση της εργασίας μου. Τέλος, ευχαριστώ ιδιαίτερα τον Επίκουρο Καθηγητή κο. Παναγιώτη Σταματόπουλο που ανέλαβε ως επιβλέπων της διπλωματικής μου εργασίας εκ μέρους του τμήματος Πληροφορικής και Τηλεπικοινωνιών του ΕΚΠΑ.

# CONTENTS

## LIST OF FIGURES

# LIST OF TABLES

# 1. BACKGROUND

## 1.1. Machine Learning

Machine learning is a subset of artificial intelligence which gives computers the ability to learn from data. The primary aim is to make predictions by learning from data, instead of being explicitly programmed by humans to do so. As a result, it enables us to tackle difficult tasks that would be impossible to be solved by fixed programs.

Machine learning algorithms can be divided into three broad categories:

- **Supervised Learning:** The input data, which are also called “training data”, have a certain label. The goal is to create a model that maps the training data to their labels, but also does so to unforeseen data.
- **Unsupervised Learning:** The algorithm has to find a structure on its own, without being given any labels.
- **Reinforcement Learning:** The program interacts with an emulated environment, by exploring the available space, in order to achieve a certain goal. Depending on its decisions, it is given a feedback score, which it aims to maximise.

## 1.2. Deep Learning

Deep learning is a subset of machine learning that imitates a human brain in processing data and creating patterns for use in decision making. For that purpose it uses networks capable of learning automatically from data which are called Artificial Neural Networks, or simply neural networks. An ANN is based on a collection of connected units or nodes called neurons. Similar to the biological brain, each connection can transmit a signal to other neurons. The connections, also called edges, typically have a weight that adjusts as learning proceeds.

Typically, neurons are organized in layers. The neurons of each layer are connected to the neurons of the next and previous layers of the network. The initial data are given to the first layer of the network, which is called the input layer. The layer that gives out the final result is the output layer. The layers between the input and the output layer are called hidden layers. Different layers may perform different kinds of transformations on their inputs.

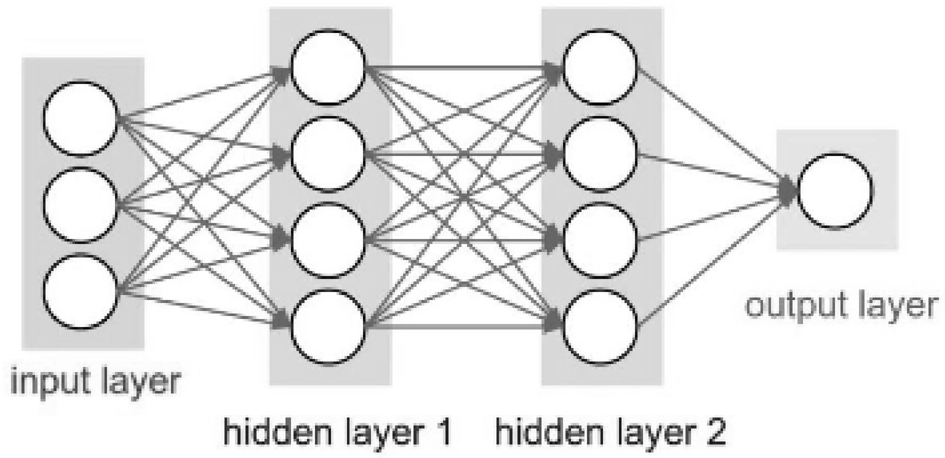

**Figure 1.1: Neural Network example**

Usually, the training dataset is passed multiple times through the network. Each cycle is referred to as an epoch. The number of epochs for which the model has a better generalization when given a new "unseen" input varys. It can depend on the size of the training data and the architecture of the neural network.

Deep learning has seen an exponential growth over the past decade due to the advances in hardware, and more specifically GPUs. It has achieved state-of-the-art results in various problems, such as computer vision and automatic speech recognition.

### 1.2.1. Neurons and layers

A single neuron is often referred to as a perceptron. As shown in Fig 1.1 it receives any number of inputs. Each input has a certain real-valued weight. The output of the neuron is a linear combination of its inputs of the form:

$$\sum_i w_i x_i + b$$

where $w_i$ is the weight of the $i$-th input, $x_i$ is the $i$-th input value and $b$ is a bias. The result of this linear combination is passed from an activation function, which we will describe later on.

A typical neural network has multiple layers, which consist of multiple neurons. Each neuron is connected to those of the next layer. A layer is called Dense or Fully connected, when it feeds all its inputs to all its neurons. Figure 1.1 shows an example of a fully connected neural network.

The input data are given to the input layer of the network. The layer that gives out the final result is called the output layer. All the layers in between are hidden layers. Different layers may perform different kinds of transformations on their inputs.

### 1.2.2. Activation function

The linear combination that is the output of the neuron is inputted to an activation function. This function enables the network to compute and represent arbitrarily complex functions. The most commonly used activation functions are nonlinear functions, as linear ones are equivalent to a neural network with a hidden plane and therefore their capacity is limited. The Sigmoid, the tanh and ReLU (Rectified Linear Unit) are some frequently used examples.

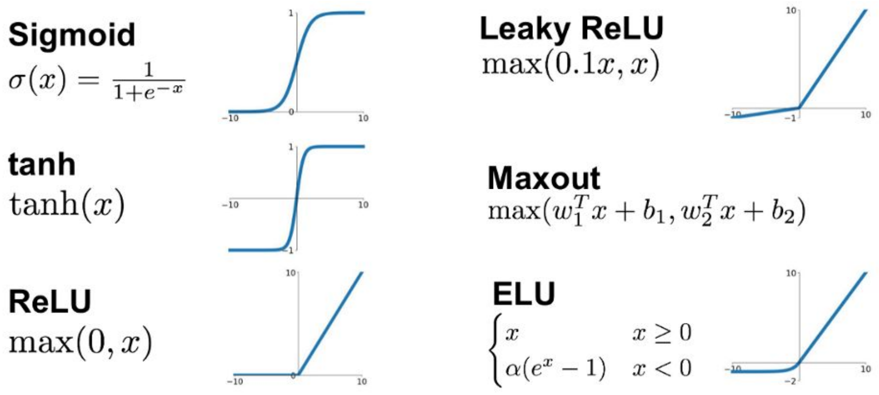

**Figure 1.2: Activation functions**

### 1.2.3. Loss function

A loss function or error function is a function that maps an event or values of one or more variables onto a real number intuitively representing some "error" associated with the event. In deep learning, it is used to express the difference between estimated and true output of a model, given an instance of data. The more the predictions deviate from the actual results, the bigger the value of the loss function. Gradually, with the help of optimization algorithms, loss function can be used to learn to reduce the error in prediction.

Loss functions can be classified into two broad categories, depending on the problem that the NN is dealing with: Regression loss and Classification loss functions. In classification, we are trying to predict output from a set of finite labels, while regression deals with predicting a continuous value for example salary or weight.

In the rest of this section, we will exhibit the two most used functions of each category.

**MSE** or Mean Square Error or simply "L2" is a regression loss function that measures the average of the squared difference between predictions and actual values.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

**MAE** or Mean Absolute Error or "L1" is a regression loss function that measures the average of the sum of the absolute differences between predictions and actual values.

$$MSE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$$

**Hinge Loss** or Multi class SVM loss is a classification loss function which aims to set a score to the correct label that is greater than the sum of scores of all the incorrect labels by some safety margin(usually one).

$$SVMLoss = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

**Cross Entropy Loss** or Negative Log Likelihood is a classification loss function Cross-entropy loss increases as the predicted probability diverges from the actual label.

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

### 1.2.4. Backpropagation

Backpropagation is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and a loss function, it calculates the gradient of the loss function depending on the neural network's weights by the chain rule.

The calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights, which is the reason why it is referred to as "backpropagation". Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow of the error information, in comparison to the naive approach of calculating the gradient of each layer separately, allows for efficient computation of the gradient at each layer.

One common algorithm that is used in order to find the set of weights that minimizes the error is gradient descent. Backpropagation is then used to calculate the steepest descent direction in an efficient way.

### 1.2.5. Gradient descent

Now that we have explained what Backpropagation is, it is clear how important gradient descent is.

Gradient descent is an optimization algorithm used to minimize the error function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. As described above, this is done by adjusting the neural networks weights.

A brief description of the algorithm can be the following. As displayed in Fig 1.3, start by choosing a random point of the function as a starting point and then calculate the derivative (gradient) at that point. If it is negative then choose another point to the right of the function relative to the value of the gradient (“step”), while if it is positive choose a point that is a step to the left. Repeat this process until either the gradient is 0 – in which case we found our minimum – or the gradient changes sign, in which case we decrease the step even more and keep moving towards the minimum.

Gradient descent with backpropagation is not guaranteed to find the global minimum of the error function, but only a local minimum. However, finding a local minimum has been proven to yield good enough results without being too computationally expensive.

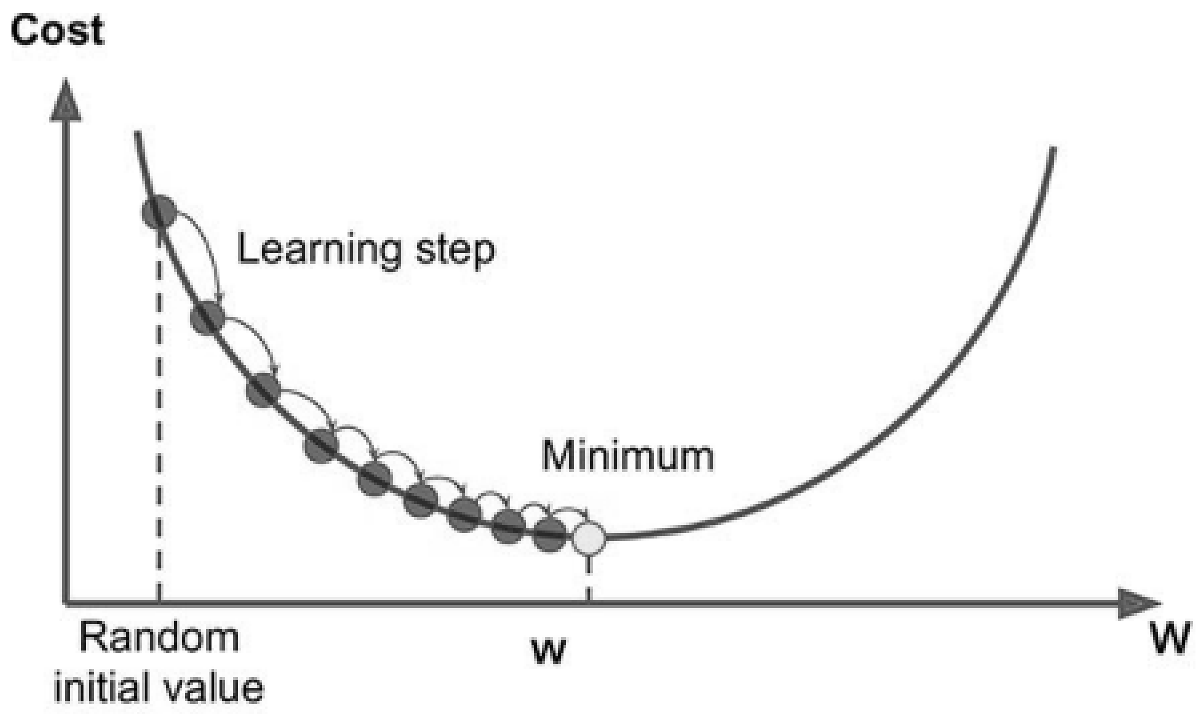

**Figure 1.3: Gradient descent graph**

## 1.3. Transfer Learning

Transfer learning is a method in machine learning that aims to make use of the knowledge gained while solving one problem and apply it to a different but similar problem. For example, the knowledge gained from training a model which has the ability to detect cars could be used to bootstrap the training of a model that detects trucks.

Transfer Learning is useful in various machine learning tasks, and especially in deep learning. In order to solve complex problems, neural networks require vast amounts of data to train from. This amount of data can be difficult to acquire, given the time and effort required to label them. For example, the ImageNet[16] dataset, which contains 14 million labeled images, was created in a long period of time (approximately 3 years) and required the participation of thousands of crowdworkers.

Creating a dataset as big as Imagenet for every specific domain is not always necessary, thanks to Transfer Learning. The knowledge from the state-of-the-art models that were trained on big datasets can be used to initialize the weights of a model that aims to solve a similar computer vision problem.

In this work, we will make use of the weights of Xception[2], which is a deep neural network that has achieved high accuracy on the ImageNet dataset.

## 1.4. Convolutional Neural Networks

Convolutional Neural Networks have been very successful at computer vision tasks. CNNs are different from simple fully-connected networks, especially towards regularization.They take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, when it comes to the scale of connectedness and complexity, CNNs are on the lower extreme.

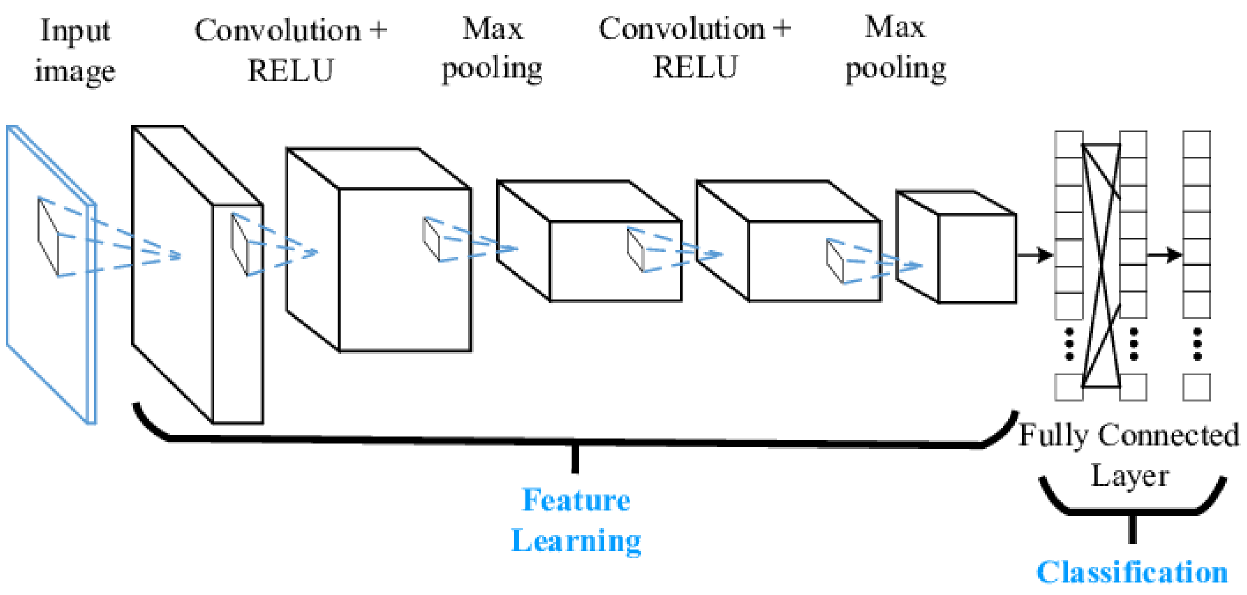

**Figure 1.4: An example convolutional neural network. From [26]**

### 1.4.1. Convolutional layer

Convolutional layers are the core building blocks of a CNN. Their parameters consist of a set of learnable filters, which are also called kernels. The innovation of CNNs is the ability to automatically learn a large number of filters, regarding the solution of a certain vision related task. The result is highly specific features that can be detected anywhere on input images.

The filters apply a dot product multiplication to the input image. They are smaller than the two-dimensional input, so that they can be applied multiple times at different points of the input. In particular, the filter is applied to each overlapping part or filter-sized patch of the input data, left to right, and top to bottom. As a result, the application of the same filter across an image allows the filter to discover whether that feature is present anywhere in the image.

Stacking convolutional layers allows a hierarchical decomposition of the input. The filters that operate directly on the raw input images extract low-level features, such as lines, while the deeper layers of the network extract higher-level features, such as shapes, houses, animals e.t.c.

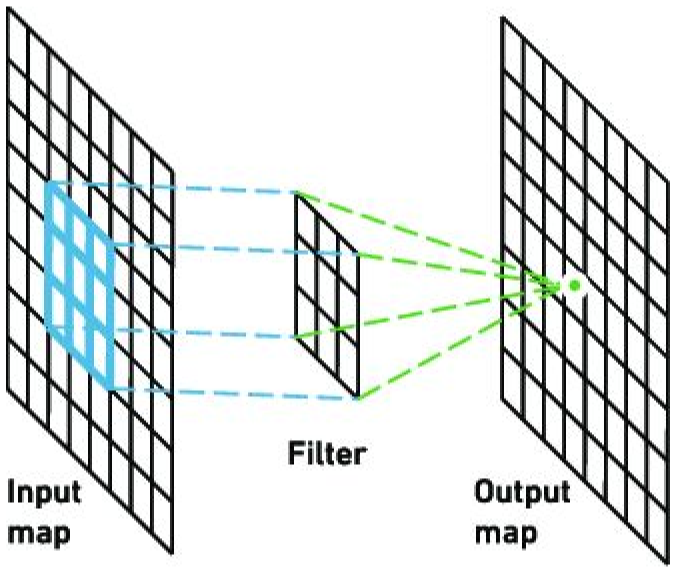

**Figure 1.5: The application of a filter in an input image**

## 1.4.2. Pooling layer

Pooling layers are equally important to CNNs. Their purpose is to down-sample their input. There are multiple nonlinear functions to implement pooling, among which max pooling is the most popular. It partitions the input image into a set of non-overlapping rectangles, and outputs the maximum of each sub-region.

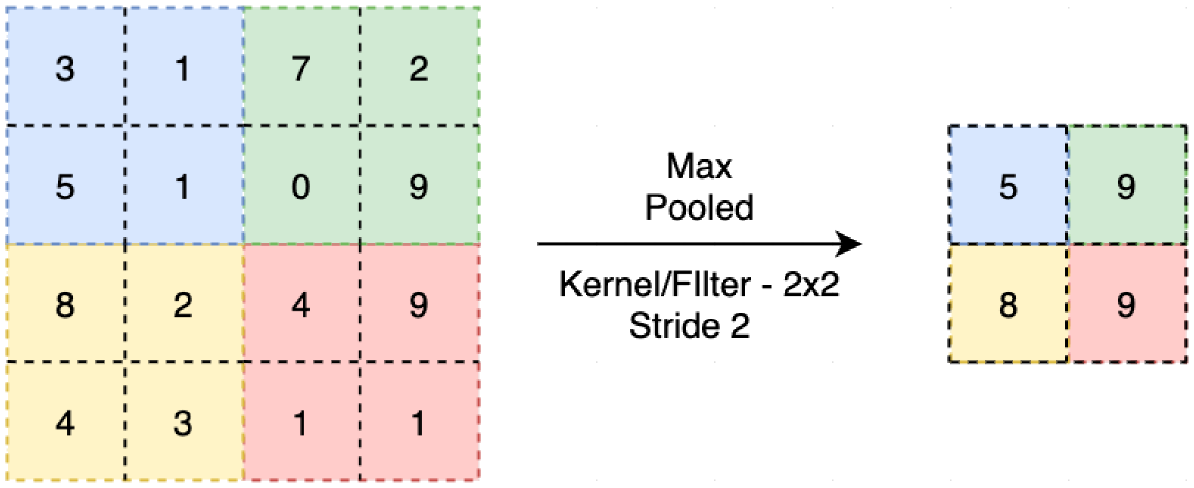

**Figure 1.6: Example of a pooling operation**

### 1.4.3. Dropout layer

From the sections above it can be inferred that machine learning models, and especially deep neural networks are very powerful models that can learn very complicated relationships between their inputs and outputs. Nevertheless, when the input set is not of sufficient size, these complicated relationships can result in sampling noise so that they can have a more precise prediction of the training set. Consequently, they may fail to make accurate predictions on real test data, even if they are drawn from the same distribution. This modeling error, that is when the model learns the detail and noise of the training data and is ineffective when it comes to new data, is called overfitting and there are multiple methods to prevent it.

Dropout[18] is one of the easiest and most commonly used methods to reduce overfitting. It refers to dropping out units of a neural network. During a training stage where dropout is adopted, the output nodes are temporarily removed with a given probability (usually 0.5). On the next epoch, the removed nodes are reinserted, along with the edges and their original weights.

Dropout aims to create a reduced network, and make the edges more independent. It is equivalent to sampling a sub-network of fewer edges. A neural network of $n$ neurons, can be seen as a collection of $2^n$ possible sub-networks. In each epoch, a new sub-network is randomly selected and trained. These networks all share weights, thus their training results in the training of the total neural network.

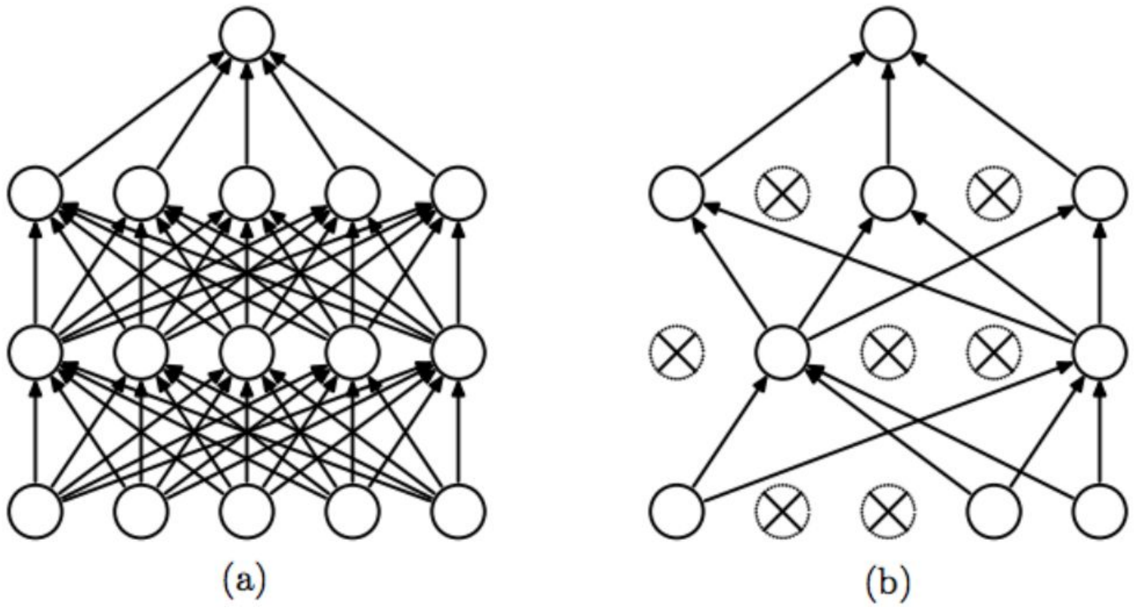

**Figure 1.7: Dropout example on a sample neural network (a) Before applying dropout (b) After applying dropout**

## 1.5. Human Action Recognition

### 1.5.1. Task definition

The task of human action recognition lies in the broader field of human centered motion recognition. A human motion recognition task can be summarised as the automatic identification of human behaviours from images or video sequences. Between several other human-centered research tasks in computer vision (e.g. human detection/tracking and pose estimation) human motion recognition is exceptionally important, duo to its multiple potential applications in video surveillance, human-computer interfaces, ambient assisted living, human-robot interaction, intelligent driving, etc.

Although the advent of low-cost RGB-D sensors and the advances in hardware such as graphics processing units (GPUs) has motivated the significant development of the task, human motion recognition remains a highly challenging task due to background clutter, partial occlusion, view-point, lighting changes, execution rate and biometric variation.

There are several tasks in the field of human centered motion recognition. Wang et al [5] has presented a broad categorization into four kinds: gesture, action, interaction and group activity.

- **Gesture recognition:** A gesture usually has a short duration and is based on the movement or positioning of a certain body part (e.g. hand, arm, body or head) that communicates an idea, an emotion, etc. Some examples of gestures are “hand waving” and “nodding”.
- **Action recognition:** Action recognition, which is also the main focus of our work, aims to identify actions. An action is considered to be a type of motion that involves multiple body parts, in comparison with gestures that involve only few. However actions are also performed in a short duration of time. “Falling”, “handshaking” and “answering the phone” are examples of human actions. When the motion is composed of a sequence of actions, then the task is called activity recognition instead.
- **Interaction recognition:** Interactions are motions that involve two actors. One actor is a human while the other can be either human or an object. Namely, the task includes human-human and human-object interaction. “Hugging each other” and “playing guitar” are examples of human-human and human-object motions respectively.
- **Group activity recognition:** Group activity is the most complicated of the four, since it includes a combination of gestures, actions and interactions. Essentially, it involves more than two humans and from zero to multiple objects. “Two teams playing basketball” and “group meeting” are examples of group activities.

In addition to this categorization, each category can be divided into two more classes, namely segmented human motion recognition and continuous/online human motion recognition. The goal of segmented motion recognition is to classify a specific sequence of video frames that contains exactly one of the predefined action labels. On the contrary, online motion recognition focuses on videos that have not been temporally trimmed. Therefore, it aims to identify the time and label of each action that the unsegmented video contains.

Our work is centered on segmented human action recognition. Therefore, our goal is the identification of human behaviours from video sequences that contain exactly one action, i.e. one motion that is performed by a single person during a short time and involves multiple body parts.

# 2. RELATED WORK

Most of the early attempts of human action recognition were based on color and texture cues.Those were not very accurate, due to the aforementioned challenges.

The development of cost-effective RGB-D sensors, such as Microsoft Kinect™ and Asus Xtion™ resulted in more effective approaches, due to the depth dimension being insensitive to illumination changes and its ability to be used to estimate 3D positions of body joints. Additionally, this field has benefited from the recent advances in hardware and more particularly in graphics processing units (GPUs) since they have enabled fast training of deep neural network architectures.

The methods that have risen to utilize RGB+D data can be broadly divided into four categories: RGB-based, depth-based, skeleton-based and RGB+D-based. In RGB-based methods, the main characteristics of focus are the shape, the color and the texture of image or video, since they are based on RGB data. In contrast, depth-based methods use the depth modality, which is more reliable for estimating body silhouette and skeleton as well as providing rich 3D structural information of the image. This is due to the depth modality being insensitive to illumination variations, and invariant to color and texture changes. Skeleton-based motion recognition methods utilize skeletial information that contain the positions of human joints. Finally, RGB+D-based methods use both the RGB and depth modalities. In this work we focus on skeleton-based, segmented motion recognition.

The different benefits of the modalities have inspired various methods. In fact, deep learning has achieved exceptional results in motion recognition. There are multiple deep learning architectures adopted in this task, such as CNNs, RNNs and others.

- **CNN-based approaches:** In order to apply CNNs on skeletal data, they must be transformed into images. Their image representation should depict the spatio-temporal alterations that take place during the sequence of skeleton frames. Various approaches achieved remarkable performance. Du et al [6] represented the skeleton sequence as a matrix by concatenating the joint coordinates in each instant and arranging those vector representations in a chronological order. The matrix is then quantified into an image and normalized to handle the variable-length problem. Wang et al. [7] proposed to encode spatio-temporal information contained in the skeleton sequence into multiple texture images, namely, Joint Trajectory Maps (JTM), by mapping the trajectories into HSV (hue, saturation, value) space. Li et al. [11] encoded the pairwise distances of skeleton joints of single or multiple subjects into texture images, namely, Joint Distance Maps (JDM) (Figure 2.1). Liu et al. [9] introduced an enhanced skeleton visualization method to represent a skeleton sequence as a series of visual and motion enhanced color images.

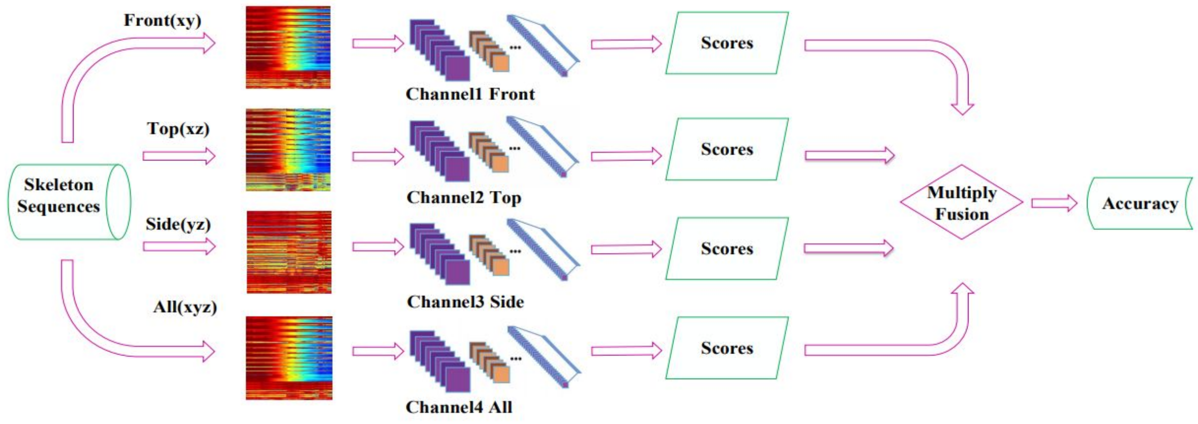

**Figure 2.1: The proposed network that uses Joint Distance Maps. From [11]**

- **RNN-based approaches:** RNNs are exceptional in exploiting the temporal information of the sequences of skeletons. Therefore, these approaches achieve better results in online human motion recognition than CNNs. Despite their success in online human motion recognition, RNNs have also accomplished competent results in the task of segmented human motion recognition as well. Du et al. [19] divided the human skeleton sequence into five parts according to the human physical structure, and separately fed them into five bidirectional RNNs/LSTMs as shown in Fig. 2.2. As the number of layers increases, the representations extracted by the subnets are hierarchically fused, resulting in a more and more higher-level representation.

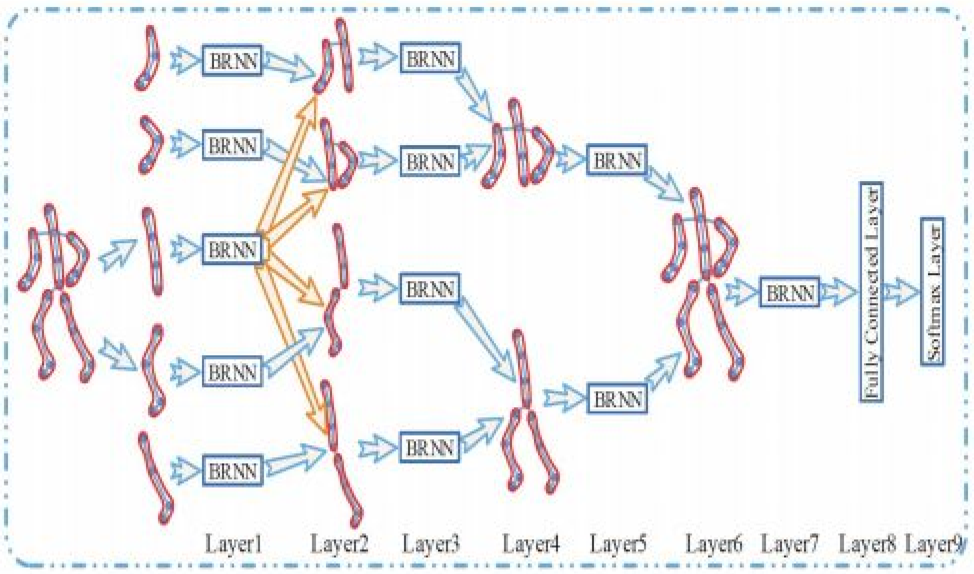

**Figure 2.2: The LSTM autoencoder and future predictor model. From [19]**

Shahroudy et al. [20] proposed a part-aware LSTM human action learning model (P-LSTM) in which the memory is split across part-based cells, showing that keeping the context of each body part independent is more efficient. Liu et al. [21] proposed a spatio-temporal LSTM (ST-LSTM) network which extends the traditional LSTM-based learning to both temporal and spatial domains. ST-LSTM explicitly models the dependencies between the joints and applies recurrent analysis over spatial and temporal domains concurrently, rather than concatenating the joint-based input features. Additionally, a trust gate mechanism to make LSTM robust to noisy input data was introduced.

- **Other-architecture-based approaches:** Apart from CNN and RNN-based architectures, there are several other methods adopted in action recognition. Salakhutdinov et al. [22] proposed a new compositional learning architecture that combines structured hierarchical Bayesian models with deep learning models. Ijjina et al. [23] adopted stacked auto-encoders to tackle the task of learning the features of input skeletal data. Huang et al. [24] incorporated the Lie group structure into a deep network architecture to learn more appropriate Lie group features for skeleton based action recognition as shown in Fig 2.3.

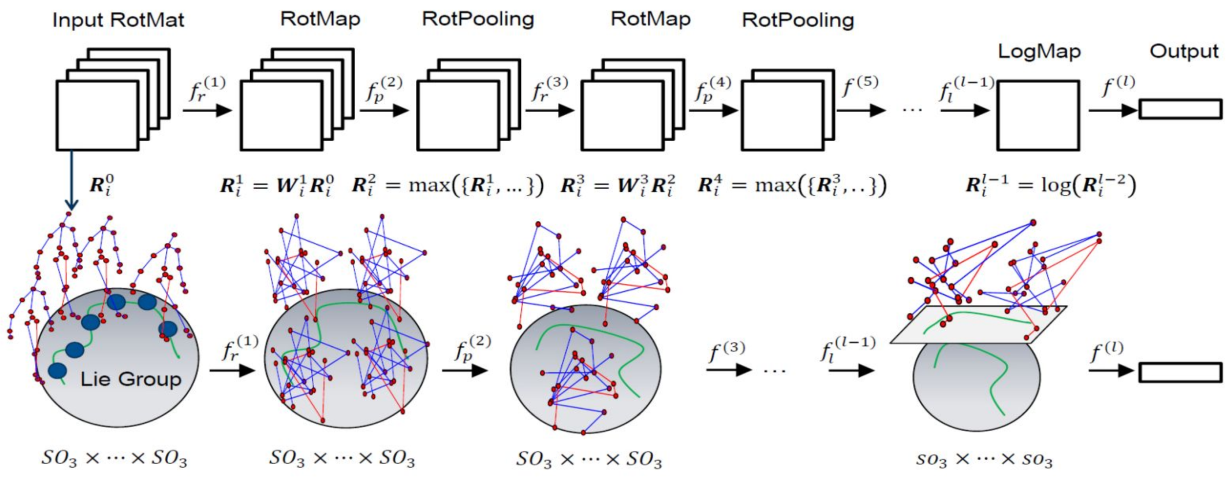

**Figure 2.3: Conceptual illustration of LieNet architecture. From [24]**

We build on works such as [12][1]. Our representation of the skeletal data is highly inspired by the latter, which proposed the use of Joint Trajectory Maps. Our networks are tuned to be effective in the more challenging task of recognizing activities of daily living instead of gaming/sport related motions. This is a harder task because the motions of the actions are less intense. Additionally, we exploit a recent state-of-the-arts architecture, the Xception[2]. Finally, as proposed by [9] we use a weighted fusion in our multi-stream CNN and receive better results than the averaging approach.

# 3. PROPOSED APPROACH

As discussed above, there are many approaches to human action recognition. In our work, we adopt the following process. First, the skeleton sequences of the PKU-MMD dataset are segmented into individual actions. Second, the actions are transformed into three JTM images. Furthermore, visual enhancements are applied to these images to draw attention to the most important motion information. Third, the images are fed into our three-stream convolutional neural network, which results in a weighted fusion. Finally the class with the highest probability is assigned to the sample. The process is explained in more detail further on.

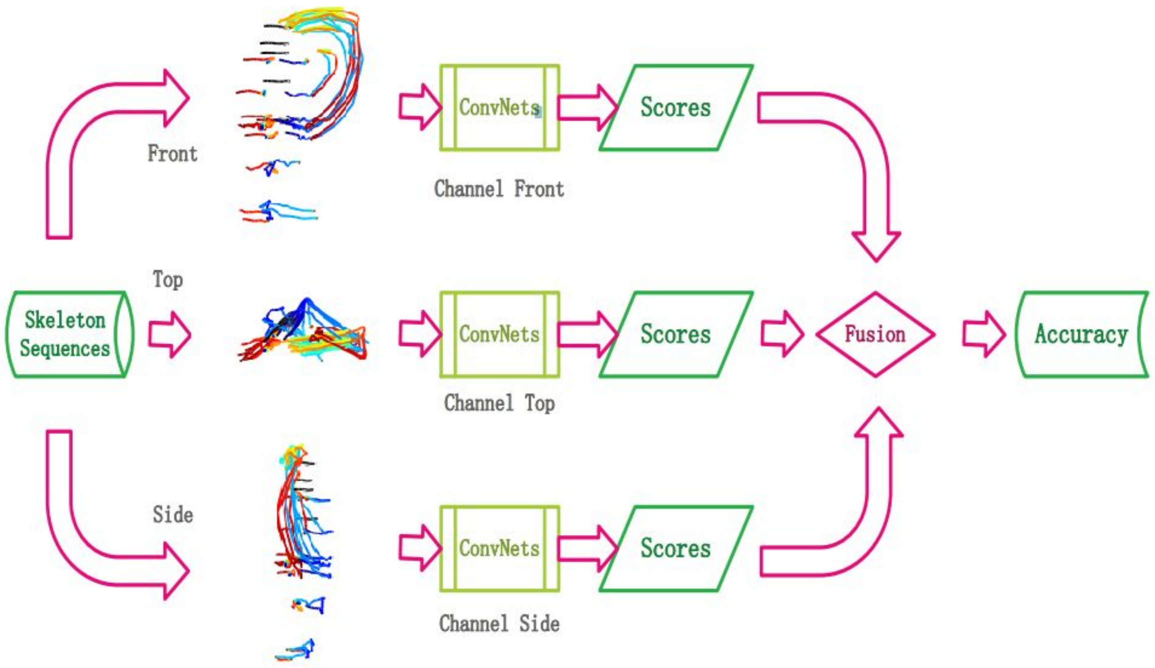

**Figure 3.1: An illustration of the proposed approach**

## 3.1. Dataset

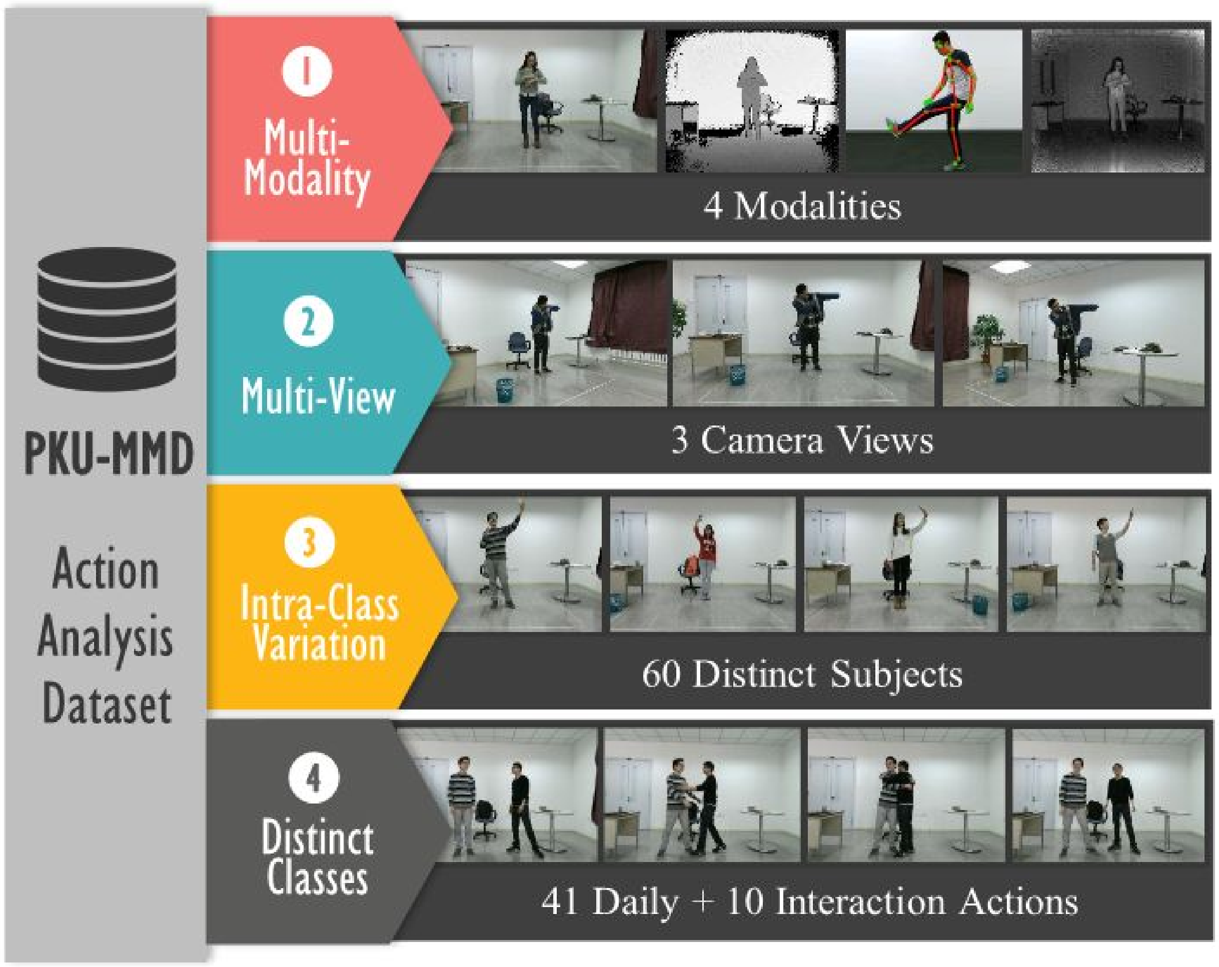

**Figure 3.2: An illustrated description of the PKU-MMD dataset. From [1]**

For our experiments we used the PKU-MMD[1] dataset. It is a collection of continuous multi-modal 3D human actions, and covers a wide variety of complex human activities.It contains 1076 long video sequences in 51 action categories, performed by 66 subjects in three camera angles (Right, Left, Middle). The cameras used are Microsoft Kinect v2 cameras in order to capture RGB-D information. The features provided are depth maps, RGB images, skeleton joints, infrared sequences, and RGB videos of the activities.

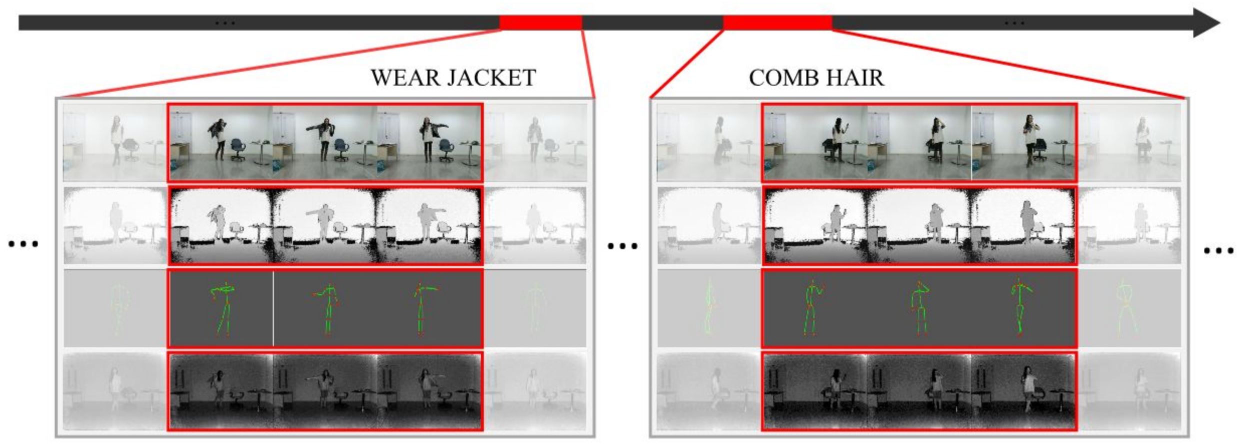

**Figure 3.3: From top to bottom, these four rows show RGB, depth, skeleton and IR modalities, respectively. From [1]**

In each video the subject performs multiple actions (approx. 20 action instances per video). The total number of actions in the dataset is approximately 20000 in 5.4 million total frames. Each action belongs to one of 51 classes (drinking, waving hand, putting glasses on, etc).

**Figure 3.4: RGB Video examples from the dataset. From [1]**

In our work, we will make use of the 3D skeleton joint positions(i.e. x,y and z coordinates) and ignore the rest of the features(RGB, depth and infrared). The dataset provides temporal information about the start and end of each action. Since our task is segmented action recognition, we use this information, to split each skeleton sequence of a video into smaller sequences that each contains an individual action.

Note that the number of frames of each skeleton sequence may vary. The reason is not only the fact that different actions may require different amounts of time, but also different subjects or even the same one may perform the same action with similar, yet not equal duration.

## 3.2. Skeletal information

A human skeleton, as presented by the Kinect Skeleton System, corresponds to a graph where nodes correspond to body parts such as arms, legs, head, neck etc. and edges follow the body structure. A skeleton consists of 25 human joints (up to 6 skeletons are simultaneously extracted in real time by using the Kinect SDK). Each joint is represented by 3 float numbers which are its 3 dimensions(x,y and z). Therefore a skeleton is an array of 75 floats.

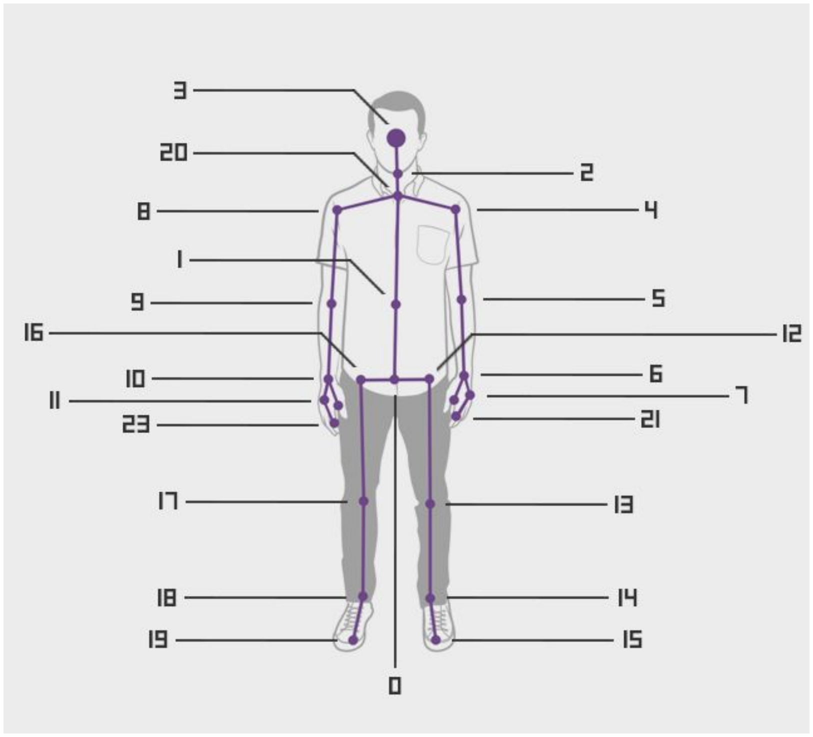

**Figure 3.5: Kinect v2 joint id map. From [25]**

## 3.3. Joint Trajectory Maps

Given a skeleton sequence, we construct three JTMs, which are then fed to the three CNNs. According to [7], an effective transformation should have the following properties:

- The joints or group of joints should be distinct in the JTM such that the spatial information of the joints is well reserved
- The JTM should encode effectively the temporal evolution, i.e. trajectories of the joints, including the direction and speed of joint motions.
- The JTM should be able to encode the difference in motion among the different joints or parts of the body to reflect how the joints are synchronized during the action

JTMs can be defined as:

$$JTM_i = JTM_{i-1} + f(i)$$

Where $f(i)$ is a function encoding the spatio-temporal information at frame $i$.

### 3.3.1. Trajectories

Let H be an action with n frames of skeletons of m joints each. Then

$$H = \{F_1, F_2, ..., F_n\}$$

where

$$F_i = \{P^i{}_1, P^i{}_2, ..., P^i{}_m\}$$

is a vector of the joint coordinates at frame $i$, and $P^i{}_j$ is the 3D coordinates of the $j$ th joint in frame $i$. The skeleton trajectory T fon an action of n frames consists of the trajectories of all joints and is defined as

$$T = \{T_1, T_2, ..., T_{n-1}\}$$

where $T_i = \{t^i{}_1, t^i{}_2, ..., t^i{}_m\} = F_{i+1} - F_i$ and the $k$th joint trajectory is $t^i{}_k = P^{i+1}{}_k - P^i{}_k$. At this stage, the function $f(i)$ is the same as $T_i$:

$$f(i) = T_i = \{t^i{}_1, t^i{}_2, ..., t^i{}_m\} .$$

Finally, the trajectories are projected to the three orthogonal planes (i.e. xy, xz, yz).

Fig. 3.6 shows the JTM of an action in the class “hand waving” of the PKU-MMD dataset, and Fig. 3.7(a),(b),(c) shows its projections to the three orthogonal planes respectively.

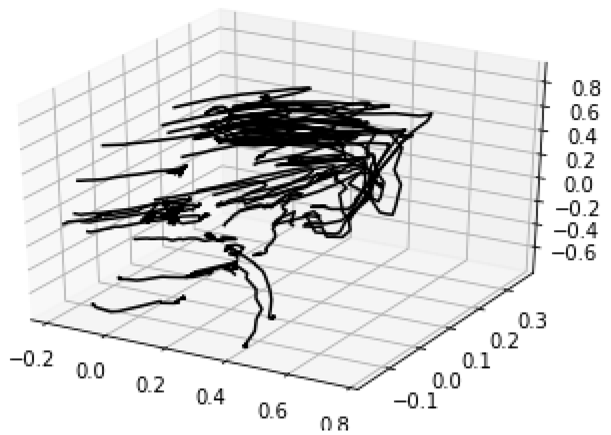

Figure 3.6: JTM of a “hand waving” action in the 3-D space

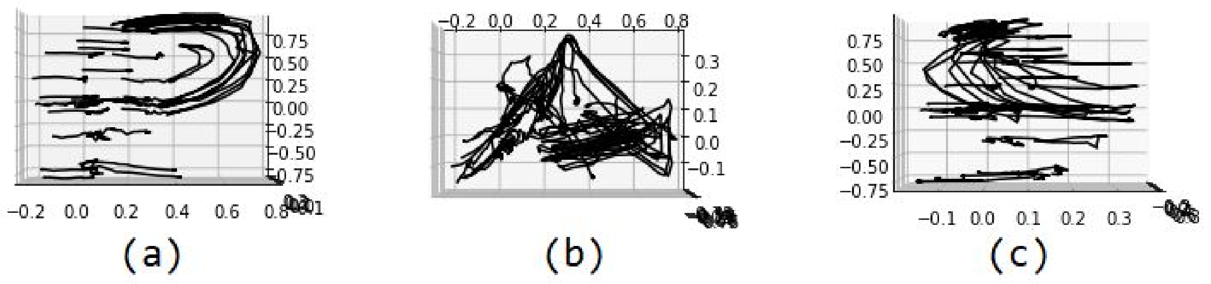

Figure 3.7: the projections of the “hand waving” action to the three orthogonal planes. (a) Projection to the x,y axis(front) (b) Projection to the x,z axis(top) (c) Projection to the y,z axis(side)

An upward movement of the left hand can be seen in Fig. 3.6 (a).

## 3.3.2. Joint Motion Direction

In order to portray the motion information of the skeleton sequence, we adjust the hue of the JTM. Using a colormap (e.x. jet colormap, ranging from blue to red) we apply it to the jtm, so that the motion is distinguishable. In particular, let the color of a joint trajectory be $C$ and the length of the trajectory L. Let $C_l, l \in (0, L)$ be the color at position $l$. The index $l$ for the $q$th trajectory will be $l = \frac{q}{n-1} \times L$. Therefore its color will be $C_l$, i.e. the color that corresponds to the index $l$ of the colormap $C$.

## 3.3.3. Body Parts

With aim to distinguish the different groups of joints, multiple colormaps are used. In our experiments, we have divided the body into three body parts. The left body part contains the left shoulder, the left elbow, the left wrist, the left hand, the left hip, the left knee, the left ankle and the left foot. The right body part consists of the right shoulder, the right elbow, the right wrist, the right hand, the right hip, the right knee, the right ankle and the right foot. The middle body part contains the rest joints, namely the head, the neck, the torso and the hip center. For each body part, we apply a different colormap. C1 is a jet colormap (Fig. 3.7 d.), and it is used for the left body part. C2 is the reverse of the jet colormap, and it is used for the right body part. Finally, C3 is a colormap ranging from white to black.

### 3.3.4. Motion Magnitude

For an action, a large motion magnitude indicates more motion information. Therefore, motion magnitude is one of the most important factors in human motion. In order to encode this information, we will adjust the saturation and brightness of each trajectory. Therefore, actions with high motion will have enriched texture, which is beneficial for the CNNs. In particular, the saturation is set to a range from $s_{min}$ to $s_{max}$. Given a trajectory, its saturation $S^i_j$ in HSV color space can be calculated as

$$S^i_j = \frac{v^i_j}{max\{v\}} \times (s_{max} - s_{min}) + s_{min}$$

where $v^i_j$ is the speed of the $j$ th joint at the $i$ th frame:

$$v^i_j = \|P^{i+1}_j - P^i_j\|_2 .$$

Thus the trajectory of joints with low motion are diluted, while those with fast motion will be saturated.

To encode the speed of the movement of the joints, the brightness of the JTM is modulated in a similar way. Specifically, the brightness is set to a range from $b_{min}$ to $b_{max}$. Given a trajectory, its brightness $B^i_j$ in HSV color space can be calculated as

$$B^i_j = \frac{v^i_j}{max\{v\}} \times (b_{max} - b_{min}) + b_{min}$$

The outcome of applying these adjustments can be seen in Fig.3.7.

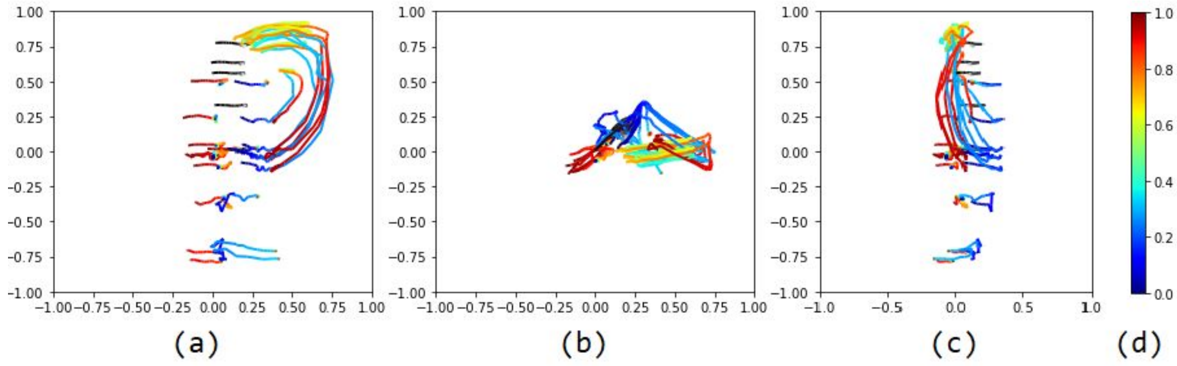

**Figure 3.8: (a),(b) and (c) are the results of applying the hue/saturation/brightness adjustments on a “hand waving” example action. (d) Jet colormap**

## 3.4. Preprocessing

Initially, the skeleton sequences of the PKU-MMD dataset have to be segmented into individual actions. The dataset provides temporal information about the start and end of each action.

To encode the spatio-temporal information, we transform each action's skeleton sequence into a JTM, which is then projected to the three orthogonal planes. The result is three 256x256 images. As described in the previous section, the three images are further enhanced by adjusting their hue,saturation and brightness. The images are normalized from [0,255] to [0,1] and fed to the CNNs.

## 3.5. Architecture

For obtaining more discriminative features from spatio-temporal skeleton joints, we adopt a multiple CNN-based model. Specifically, our model consists of three streams of CNNs. Each stream corresponds to one of the three projections of the JTMs (i.e. front, top, side). The scores generated from each stream are fused to compute the final score.

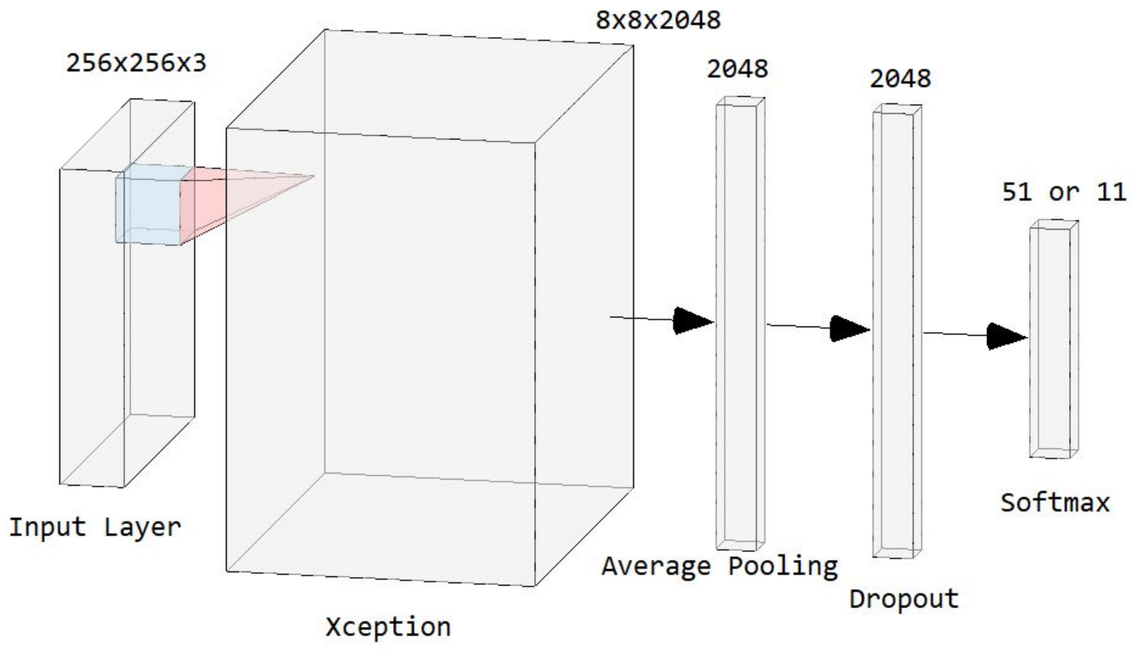

**Figure 3.9: A single stream of the total neural network**

As depicted in Fig. 3.9, the complete network has three inputs, which correspond to the front, top, side CNN respectively. Given as input the three JTMs images, each image is passed to the corresponding CNN. Each stream contains 126+3 layers(Fig. 3.8.). The input is passed to an Xception model, which has 126 layers. This model appeared to have the best results in our problem, compared to VGG and ResNet50. Its output is connected to an 2D Average Pooling layer which gives out a tensor of shape (2048). Next is a dropout layer of rate 0.5. The output of the dropout layer is fed to a 51-way (or 11-way) softmax which produces the distribution over the 51(or 11) class labels. The fusion method is discussed in the following section.

Inspired by [10], we experimented with two late fusion methods: adding (which is similar to averaging, but appeared to be slightly better) and training a multi-class linear SVM.

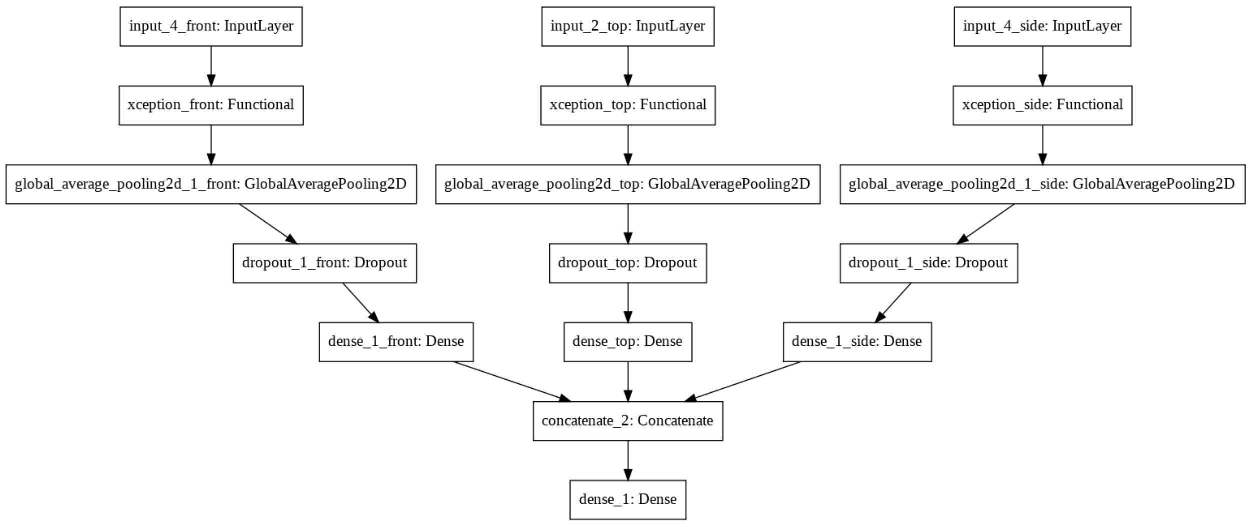
input_4_front: InputLayer
input_2_top: InputLayer
input_4_side: InputLayer
xception_front: Functional
xception_top: Functional
xception_side: Functional
global_average_pooling2d_1_front: GlobalAveragePooling2D
global_average_pooling2d_top: GlobalAveragePooling2D
global_average_pooling2d_1_side: GlobalAveragePooling2D
dropout_1_front: Dropout
dropout_top: Dropout
dropout_1_side: Dropout
dense_1_front: Dense
dense_top: Dense
dense_1_side: Dense
concatenate_2: Concatenate
dense_1: Dense

**Figure 3.10: A plot of the complete three-stream network**

## 3.6. Data Augmentation

In order to minimize overfitting, data augmentation[13] is employed. Image data augmentation or manipulation is a technique that is used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. Thus the size of the input data increases, which can result in improvement of the model's ability to generalize.

There are several ways to manipulate an image, however the technique used must be meaningful to the format of the data. In our work, the technique that resulted in better performance was a random horizontal flip of the images.

The transformation of the original images takes very little computation, so the transformed images do not need to be stored on disk. In our implementation, the transformed images are generated with the use of the tensorflow.image[14] python library, which runs on the CPU while the GPU is training on the previous batch of images. As a result, the total training time is not affected.

Note that only the horizontal flip of the front and top images is beneficial. Flipping the side images would confuse the model in vain, since all the subjects are facing in the same direction when performing an action.

Additionally, in case of flipping and image, $C_1$ and $C_2$ colormaps must be swapped as well. Otherwise, the colors would be on the opposite side of the body than they are supposed to be.

## 3.7. Training Strategy

The implementation is based on Keras[15] and Tensorflow [14] on top of an NVIDIA Tesla K80 GPU card and 12GB RAM.The training procedure can be divided into 2 steps.

Firstly, the three CNNs are trained individually on their corresponding training set -i.e. front,top or side. Each CNN employs the following method. Initially, the weights for the Xception[2] layers which are pre-trained over ImageNet[16] are loaded. The model will pass the training procedure twice. In the first training, the weights of the pre-trained model are frozen. Thus, only the top layers of the CNN (AvgPool, Dropout, Dense) are trained, without destroying the information that it contains. The network weights are learnt using the Adam optimization, with learning rate equal to $10^{-3}$ . At each iteration, a batch of 32 samples is constructed by sampling 32 training JTMs uniformly across the classes. The images undergo random horizontal flipping as described in the data augmentation section above. In the second training session the Xception layers are set to trainable(unfrozen), so that the entire model is trained. In this step we set a low learning rate ( $10^{-5}$ ) in order to prevent it from overfitting.

Now that the three CNNs are trained the next step is to construct the three-stream model, by merging them with our fusion technique. In the case of weighted fusion, the CNNs weights are frozen in order to train the fusion layers. For the averaging fusion approach no training is required.

# 4. RESULTS

## 4.1. Metrics

Before presenting the results themselves, it is necessary to briefly explain the metrics used. In order to evaluate the effectiveness of the classification task throughout our experiments we used various metrics, such as Accuracy, Confusion Matrices, Precision and Recall.

- **Accuracy:** Classification accuracy by definition, is the ratio of number of correct predictions made by our model to the total number of input samples. In other words, it expresses how many skeleton sequences were classified to their correct action label by our method, relative to the total number of actions of the training set. It is the most commonly used way of assessing how well a classification model performs.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made}$$

- **Confusion Matrix:** A confusion matrix, or error matrix, is an N x N table that visualizes the performance of a machine learning model, where N is the number of classes. More specifically, it compares the actual target labels with those predicted by the model. This way it demonstrates how well the classification performed and what kinds of errors were made.

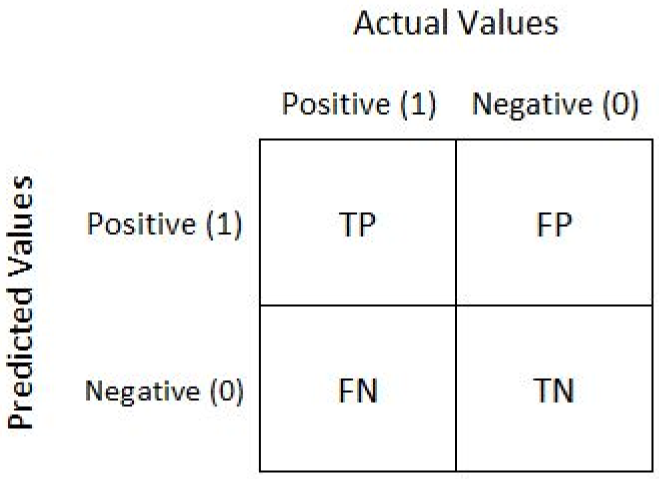

**Figure 4.1: Confusion matrix**

In order to understand confusion matrices, one needs to know what are True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) in a binary classification task. Let every sample of an input set have two possible labels: positive or negative. A sample is classified as True Positive when its predicted label is positive and its actual label is also positive, therefore the predicted label matches the actual label. True Negatives' predicted label also matches the actual label, but that label is negative. A False Positive is a sample that was falsely predicted; it's actual label was negative but the model predicted the positive label. Similarly, a False Negative is also mispredicted, but its actual label is positive while the model predicted the negative label. The corresponding table that shows these values, is called the confusion matrix.

- **Precision:** Precision informs us about how many of the correctly predicted samples actually turned out to be positive. It is calculated by the following fraction:

$$Precision = \frac{TP}{TP + FP}$$

- **Recall:** Recall is about how many of the actual positive samples were able to be predicted correctly by our model. It is calculated by the following fraction:

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score:** The f1-score is the harmonic mean of Precision and Recall. When we want to increase the precision of our model, recall decreases and vice-versa, therefore the f1-score is a helpful metric since it captures the trends between them in a single value.

$$f1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

- **Support:** Support is the actual number of samples of each class.

## 4.2. Test cases

The proposed method is evaluated on the PKU-MMD dataset [1]. The process is divided into two phases. In the first part, we performed experiments with the whole dataset, i.e. using all the 51 classes. In the second phase our goal was to assess whether the proposed approach may be used for ambient assisted living scenarios and more specifically for the recognition of ADLs. Therefore, we selected 11 out of the 51 classes of PKU-MMD, which we believe are the closest to the categories presented in [8]. The selected classes are: “eat meal snack”, ”falling”, ”handshaking”, ”hugging other person”, ”make a phone call/answer phone”, ”playing with phone tablet”, ”reading”, ”sitting down”, ”standing up”, ”typing on a keyboard” and ”wearing a jacket”.

In each phase, we evaluate the models using cross-subject and cross-view splitting of the data. Cross-subject split uses different humans(subjects) to train the model than to evaluate it, and cross-view uses different cameras (e.x. train on the Left and Right camera, and test on the middle). Since the dataset is well annotated and is large with flexible actions and actors, we split the actions according to the same Cross-Subject and CrossView participation as its defined in [1].

Moreover, we present the results of using 1 channel CNNs compared to the triple stream approach, in order to showcase the improvement in accuracy. In particular, we compare the efficiency of using only the front, only the top, only the side or all the three JTM projections to draw conclusions about the effectiveness of using a fused model.

## 4.3. Results

### 4.3.1. Phase 1 - 51 classes

As mentioned above, in this phase we work with the whole dataset. This phase is more demanding since, as described in detail in section 3.1, the dataset contains approximately 20000 actions in 5.4 million total frames belonging to the 51 classes. We set our models to predict instances in 51 classes, by setting the last layer's (softmax) size to 51, and train them accordingly.

**Table 1: Accuracy of Phase 1**

| **Model** | **Cross-View** | **Cross-Subject** |
|---|---|---|
| Front only | 0.5869 | 0.6066 |
| Top only | 0.4268 | 0.4549 |
| Side only | 0.6188 | 0.6150 |
| Fusion | **0.6586** | **0.6780** |

As it may be observed from Table 1, the fusion of the three CNNs results in a great improvement of the model's accuracy. More particularly, it is more accurate by ~4% in the cross-view, and ~6% in the cross subject experiment. Consequently, the three streams are complementary to each other, and their fusion of the 3 networks was, as expected, the most successful approach.

We also notice that the Side model is more accurate, compared to the Front and the Top ones. Since their architecture is the same, that means that the Side representations depicts the actions in a more useful way for the CNNs. This might be due to the fact that the actions of this dataset are easier to distinguish when looked from the side. On the other hand, the Top CNN's accuracy is low in comparison to the other two, which means that the (yz) projections of the JTMs of different actions are hardly distinguishable.

The fact that the cross-subject scores are better than the cross-view ones, show that it is harder for the model to generalize to a new view than to a new subject.

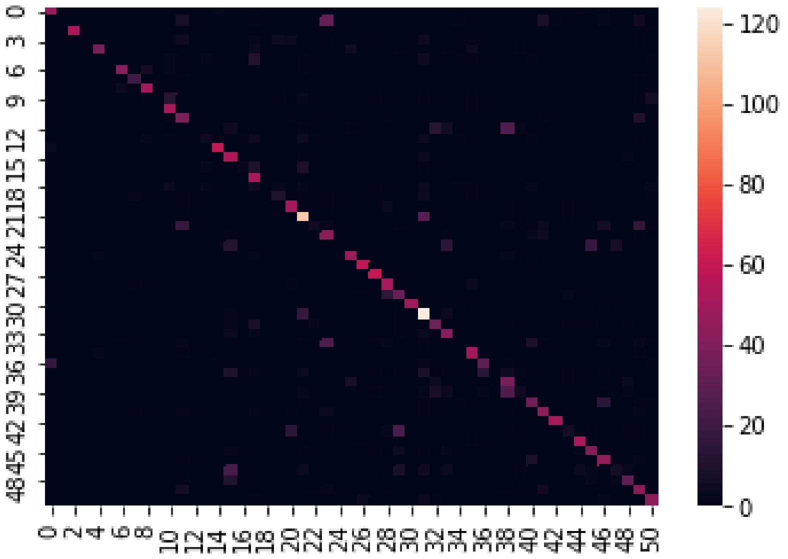

**Figure 4.2: Phase 1 Confusion matrix heatmap**

In Figure 4.2 it is noticeable that the diagonal contains brighter colors than the rest of the matrix. This was expected since we have achieved an accuracy good enough so that most samples are correctly classified. To illustrate, the diagonal corresponds to the True positives and True negatives in the example of a binary classification, and therefore contains the number of samples whose predicted label matches their actual label.

The figure shows that there exist actions that are easily distinguishable by our model, but there are also other actions that are hardly predicted. For example, the cell (31,31) is distinctly bright, therefore it can be inferred that the model can distinguish the action "rub two hands together" with a high accuracy. In contrast, the cell (24,24) seems dark, which shows that our model performs poorly on the recognition of the action "pointing to something with finger".

It can also be seen that there exist cells that do not belong to the diagonal but have a light color. Those cells indicate errors that the model often makes. The fact that the color of a cell (x,y) is bright means that the number of mispredicted samples that were classified to the class y but in reality belong in the class x is high. Those sets of classes are hardly distinguishable by the model. For instance, the class #23 is often misclassified to the class #1, since the cell (23,1) is bright. In other words, the action “salute” is often mistaken to be the action “brushing hair”. Similarly, “taking a selfie” is often mistaken with “hand waving” and “tear up paper”.

Finally, it can be seen that multiple cells are symmetric to the diagonal. For example, cells (30,21) with (21,30), and (33,24) with (24,33) .This indicates that the classification of samples that belong to either one of the classes can be misclassified to the other class.

**Table 2: Phase 1 classification report**

| # | Action label | Precision | Recall | f1-score | Support |
|---|---|---|---|---|---|
| 0 | bow | 0.68 | 0.92 | 0.78 | 51 |
| 1 | brushing hair | 0 | 0 | 0 | 57 |
| 2 | brushing teeth | 1 | 0.98 | 0.99 | 54 |
| 3 | check time (from watch) | 0 | 0 | 0 | 24 |
| 4 | cheer up | 0.91 | 0.74 | 0.81 | 53 |
| 5 | clapping | 0 | 0 | 0 | 24 |
| 6 | cross hands in front (say stop) | 0.91 | 0.82 | 0.87 | 51 |
| 7 | drink water | 0.84 | 0.88 | 0.86 | 24 |
| 8 | drop | 0.85 | 0.93 | 0.88 | 54 |
| 9 | eat meal/snack | 0 | 0 | 0 | 27 |
| 10 | falling | 0.68 | 0.96 | 0.79 | 54 |
| 11 | giving something to other person | 0.47 | 0.72 | 0.57 | 54 |
| 12 | hand waving | 0 | 0 | 0 | 54 |
| 13 | handshaking | 0.45 | 0.21 | 0.29 | 24 |
| 14 | hopping (one foot jumping) | 0.95 | 0.95 | 0.95 | 63 |
| 15 | hugging other person | 0.43 | 0.9 | 0.58 | 60 |
| 16 | jump up | 0.67 | 0.08 | 0.15 | 24 |
| 17 | kicking other person | 0.49 | 1 | 0.65 | 51 |
| 18 | kicking something | 0 | 0 | 0 | 24 |
| 19 | make a phone call/answer phone | 0.59 | 0.42 | 0.49 | 24 |
| 20 | pat on back of other person | 0.71 | 0.88 | 0.79 | 57 |
| 21 | pickup | 0.69 | 0.79 | 0.74 | 143 |
| 22 | playing with phone/tablet | 0.42 | 0.09 | 0.15 | 54 |
| 23 | point finger at the other person | 0.39 | 0.8 | 0.52 | 54 |
| 24 | pointing to something with finger | 0 | 0 | 0 | 51 |
| 25 | punching/slapping other person | 0.77 | 0.89 | 0.83 | 54 |

| | | | | | |
|---|---|---|---|---|---|
| 26 | pushing other person | 0.89 | 0.98 | 0.94 | 60 |
| 27 | put on a hat/cap | 0.91 | 1 | 0.95 | 60 |
| 28 | put something inside pocket | 0.67 | 0.89 | 0.77 | 57 |
| 29 | reading | 0.43 | 0.65 | 0.52 | 51 |
| 30 | rub two hands together | 0.98 | 0.96 | 0.97 | 51 |
| 31 | salute | 0.61 | 0.84 | 0.71 | 147 |
| 32 | sitting down | 0.57 | 0.65 | 0.61 | 54 |
| 33 | standing up | 0.55 | 0.82 | 0.66 | 51 |
| 34 | take off a hat/cap | 0 | 0 | 0 | 48 |
| 35 | take off glasses | 0.81 | 0.93 | 0.86 | 54 |
| 36 | take off jacket | 0.65 | 0.57 | 0.61 | 54 |
| 37 | take out something from pocket | 0 | 0 | 0 | 51 |
| 38 | taking a selfie | 0.36 | 0.7 | 0.48 | 54 |
| 39 | tear up paper | 0.4 | 0.08 | 0.13 | 51 |
| 40 | throw | 0.57 | 0.65 | 0.61 | 54 |
| 41 | touch back (backache) | 0.65 | 0.75 | 0.7 | 57 |
| 42 | touch chest (stomach ache/heart pain) | 0.98 | 1 | 0.99 | 51 |
| 43 | touch head (headache) | 0.46 | 0.12 | 0.19 | 51 |
| 44 | touch neck (neckache) | 0.88 | 0.96 | 0.92 | 54 |
| 45 | typing on a keyboard | 0.63 | 0.76 | 0.69 | 54 |
| 46 | use a fan (with hand or paper)/feeling warm | 0.6 | 0.79 | 0.68 | 57 |
| 47 | wear jacket | 0.4 | 0.11 | 0.17 | 57 |
| 48 | wear on glasses | 0.74 | 0.65 | 0.69 | 48 |
| 49 | wipe face | 0.49 | 0.73 | 0.59 | 60 |
| 50 | writing | 0.71 | 0.81 | 0.76 | 54 |
| | | | | | |
| | **accuracy** | 0.65 | 0.65 | 0.65 | 0 |
| | **macro avg** | 0.55 | 0.6 | 0.55 | 2704 |
| | **weighted avg** | 0.58 | 0.65 | 0.59 | 2704 |

Table 2 is an analytical classification report that shows the Precision, Recall, f1-score and Support scores for each class. By considering its values, we can verify the conclusions we made above, when we examined the confusion matrix, and make new ones.

Firstly, notice that the action “rub two hands together” has an f1-score of 0.97. This verifies the brightness of the cell (30, 30) being high, as we discussed above. We also notice that there exist classes with an f1-score of 0, which justifies the multiple dark cells in the diagonal of the confusion matrix.

Notice that there exist classes that have a high f1-score but are not very bright in the matrix. This is due to the f1-score being different from the accuracy score. These two metrics should not be assumed to be equal.

Another aspect that can be observed is the difference between precision and recall of each class. The fact that these two scores are inversely proportional, means that the recognitions of actions where this difference is small are more robust, and the recognition of actions where this difference is high can be improved.

### 4.3.2. Phase 2 - 11 classes

During phase 2, we focus on 11 out of the 51 classes. Again, we tune our model to predict in 11 classes by setting the size of the softmax layer as 11. Also we remove the actions that belong to the rest 40 classes, from the training and test sets.

In this phase we expect significantly better accuracy than Phase 1, since the 11 most suitable daily activities are selected and machine learning models are better in identifying classes that are less in quantity.

**Table 3: Accuracy of Phase 2**

| **Model** | **Cross-View** | **Cross-Subject** |
|---|---|---|
| Front only | 0.8284 | 0.8931 |
| Top only | 0.6804 | 0.7301 |
| Side only | 0.8446 | 0.9094 |
| Fusion | **0.8691** | **0.9212** |

As expected, the overall accuracy of phase 2 is much greater than that of phase 1, due to the smaller number of classes. We notice similar behaviour with phase 1 between the Front, Top and Side CNNs, as the Side is the best and the Top is the worst. Of course, the cross-subject experiment being more accurate than the cross-view one also applies in this phase. Overall, the results show that our attempt to assess whether the proposed approach may be used for the recognition of ADLs was successful.

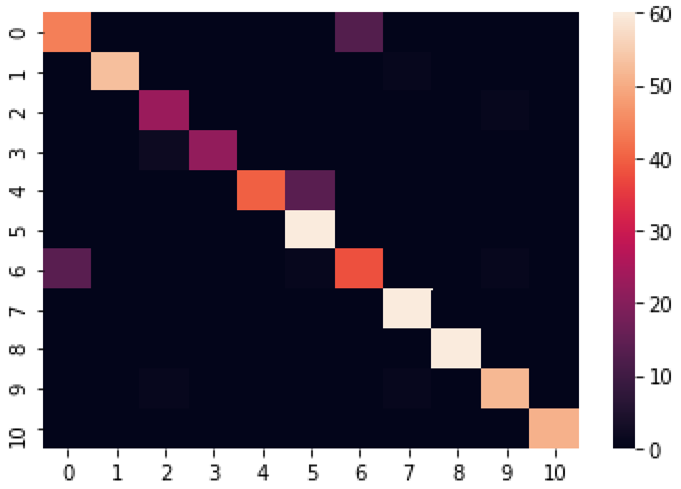

**Figure 4.3: Phase 2 Confusion matrix heatmap**

Figure 4.3 depicts the confusion matrix of Phase 2. As expected, since we have achieved an accuracy good enough so that most samples are correctly classified, the diagonal contains brighter colors than the rest of the matrix. In comparison to the previous phase, the diagonal is even brighter, since a higher accuracy was achieved in phase 2. In other words, the number of samples whose predicted label matches the actual label is relatively high.

It can be inferred that there exist actions that are easily distinguishable by our model, compared to other actions that are hardly predicted. For example, the cell (5,5) has a very high brightness, in contrast with (2,2) which is darker. Therefore it can be inferred that the model can distinguish the action “playing with phone tablet” with a high accuracy. In contrast, our model performs poorly on the recognition of the action “handshaking”.

Though, there exist cells that do not belong to the diagonal but have a light color, they are percentagewise less than those observed in phase 1. The fact that (6,0) and (0,6), which are symmetric to the diagonal, are not dark indicates that the number of mispredicted samples that were classified to class 6 but actually belonged to class 0, and vice versa, is high. Thus we conclude that the actions “eat meal snack” and “reading” are hardly distinguishable by our model.

Another point that can be made is that “playing with phone/tablet” is often mistaken to be the action “make a phone call/answer phone”. This can be inferred from the fact that (5,4) is not dark. However this is an example where the reverse does not occur; the cell (4,5) is dark.

**Table 4: Phase 2 classification report**

| # | Classes | Precision | Recall | f1-score | Support |
|---|---|---|---|---|---|
| 0 | eat meal/snack | 0.76 | 0.77 | 0.77 | 57 |
| 1 | falling | 1 | 0.98 | 0.99 | 54 |
| 2 | handshaking | 0.88 | 0.96 | 0.92 | 24 |
| 3 | hugging other person | 1 | 0.92 | 0.96 | 24 |
| 4 | make a phone call/answer phone | 1 | 0.74 | 0.85 | 54 |
| 5 | playing with phone/tablet | 0.8 | 1 | 0.89 | 60 |
| 6 | reading | 0.75 | 0.7 | 0.72 | 54 |
| 7 | sitting down | 0.97 | 1 | 0.98 | 60 |
| 8 | standing up | 1 | 1 | 1 | 60 |
| 9 | typing on a keyboard | 0.96 | 0.96 | 0.96 | 54 |
| 10 | wearing a jacket | 1 | 1 | 1 | 51 |
| | | | | | |
| | **accuracy** | 0.91 | 0.92 | 0.92 | 0 |
| | **macro avg** | 0.92 | 0.92 | 0.92 | 552 |
| | **weighted avg** | 0.92 | 0.92 | 0.92 | 552 |

The first characteristics that can be observed about the classification report of Table 4 in comparison to that of Phase 1, are the higher scores and the smaller difference between the precision and recall scores of each class. This is a result of the smaller number of classes that this phase experiments with.

In summary, our experiments have shown that our JTM transformation method captures the spatio-temporal information of the skeleton sequences effectively and that our model is capable of exploiting this representation with aim to successfully recognize human actions. The results validate that our approach is potent and that it can be used for the recognition of ADLs in a real life scenario.

# 5. CONCLUSION

## 5.1. Conclusion

In this thesis we presented a novel approach which aims to recognize human actions in videos. Our approach is based on Joint Trajectory Maps and CNNs. Initially, since the task we tackled is segmented action recognition, we had to split the activities of the dataset into individual actions. We have presented the process of the construction of JTMs, when a skeleton sequence is given as input. The spatio-temporal information of the 3-D skeleton sequences are encoded into JTMs. Additional properties of the input images are exploited, such as hue/brightness and saturations, in order to illustrate the skeletal information in an as detailed as possible manner. We have demonstrated the architecture of our multi-stream Convolutional Neural Network, and described the training process.

The results of our evaluations show that Joint Trajectory Maps can be an effective technique to encode spatio-temporal information, and their combination with deep convolutional neural networks is very promising in the task of human action recognition. Even on a challenging dataset like PKU-MMD, we have achieved competent results.

Last but not least, the evaluation process is fast enough to be used for real-time classification, in a real-world scenario (0.58 seconds/sample). However, one thing that must be noted at this point is that real-life-like environments are not as ideal as the samples of most datasets. For example, most available datasets do not involve much occlusion cases probably due to the collapse of skeleton information during the occlusion. However, in practical scenarios, occlusion is inevitable.

## 5.2. Future Work

The method we have adopted uses one type of neural networks. Its input consists exclusively of Joint Trajectory Maps. A future direction for research could be to attempt to create hybrid networks that exploit multiple modalities. For example a fourth stream could be added to the network, which consists of the model from previous research [12]. Alternatively the model could also exploit the RGB and depth modalities which are provided by the dataset. This way the network is provided with more diverse information and can result in better generalization.

Another drawback of our approach is that it relies on strictly segmented data. In general, the capability of online recognition is quite limited. In real-world applications, the segmentation of the data has to happen automatically as well. This work can be extended with the use of RNNs for the purpose of temporal segmentation.

As discussed above, the gap between the available datasets, and the real world is big. Another interesting direction that can be followed is to investigate how applicable our methods are to the real world, where challenges like background clutter, partial occlusion, view-point, lighting changes, execution rate and biometric variation occur. Then, the next question is how to recover from those obstacles or find cues from multi-modal data for such recognition tasks. Are the data provided by our available cameras like Kinect enough to overcome these challenges?

In summary, we managed to present an efficient approach towards the recognition of human actions, and achieved tangible results using deep learning techniques. We consider the outcome of this dissertation to be as successful as our expectations were met. We hope our experiments were insightful and will inspire future work.

# ABBREVIATIONS - ACRONYMS

| | |
|---|---|
| ANN | Artificial Neural Network |
| CNN | Convolutional Neural Network |
| ReLU | Rectified Linear Unit |
| 3-D | Three dimensional |
| JTM | Joint Trajectory Map |
| RGB | Red Green Blue |
| RGB-D | Red Green Blue and Depth |
| JDM | Joint Distance Map |
| GPU | Graphics processing unit |
| SDK | Software development kit |

## REFERENCES

[1] Liu C, Hu Y, Li Y, Song S, Liu J. Pku-mmd: A large scale benchmark for continuous multi-modal human action understanding. arXiv preprint arXiv:1703.07475. 2017 Mar 22.

[2] Chollet F. Xception: Deep learning with depthwise separable convolutions. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 1251-1258).

[3] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence. 2013 Mar 7;35(8):1798-828.

[4] Ji S, Xu W, Yang M, Yu K. 3D convolutional neural networks for human action recognition. IEEE transactions on pattern analysis and machine intelligence. 2012 Mar 6;35(1):221-31.

[5] Wang P, Li W, Ogunbona P, Wan J, Escalera S. RGB-D-based human motion recognition with deep learning: A survey. Computer Vision and Image Understanding. 2018 Jun 1;171:118-39.

[6] Du Y, Fu Y, Wang L. Skeleton based action recognition with convolutional neural network. In2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR) 2015 Nov 3 (pp. 579-583). IEEE.

[7] Wang P, Li Z, Hou Y, Li W. Action recognition based on joint trajectory maps using convolutional neural networks. InProceedings of the 24th ACM international conference on Multimedia 2016 Oct 1 (pp. 102-106).

[8] M. P Lawton and E. M Brody, “Assessment of older people: self-maintaining and instrumental activities of daily living” in The gerontologist, 9(3 Part 1), pp. 179-186, 1969.

[9] Liu, M., Liu, H., Chen, C., 2017b. Enhanced skeleton visualization for view invariant human action recognition. Pattern Recognition 68, 346–362

[10] K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, in: NIPS, 2014, pp. 568–576.

[11] Li, C., Hou, Y., Wang, P., Li, W., 2017a. Joint distance maps based action recognition with convolutional neural networks. IEEE Signal Processing Letters 24, 624–628.

[12] A. Papadakis, E. Mathe, I. Vernikos, A. Maniatis, E. Spyrou and Ph. Mylonas, Recognizing Human Actions using 3D Skeletal Information and CNNs. In Proc. of Int’l Conf. on Engineering Applications of Neural Networks (EANN), 2019.

[13] Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. Journal of Big Data. 2019 Dec 1;6(1):60.

[14] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M. Tensorflow: A system for large-scale machine learning. In12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16) 2016 (pp. 265-283).

[15] Gulli A, Pal S. Deep learning with Keras. Packt Publishing Ltd; 2017 Apr 26.

[16] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In2009 IEEE conference on computer vision and pattern recognition 2009 Jun 20 (pp. 248-255). Ieee.

[17] Hou Y, Li Z, Wang P, Li W. Skeleton optical spectra-based action recognition using convolutional neural networks. IEEE Transactions on Circuits and Systems for Video Technology. 2016 Nov 14;28(3):807-11.

[18] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research. 2014 Jan 1;15(1):1929-58.

[19] Du Y, Wang W, Wang L. Hierarchical recurrent neural network for skeleton based action recognition. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 1110-1118).

[20] Shahroudy A, Liu J, Ng TT, Wang G. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 1010-1019).

[21] Liu J, Shahroudy A, Xu D, Wang G. Spatio-temporal lstm with trust gates for 3d human action recognition. InEuropean conference on computer vision 2016 Oct 8 (pp. 816-833). Springer, Cham.
[22] Salakhutdinov R, Tenenbaum JB, Torralba A. Learning with hierarchical-deep models. IEEE transactions on pattern analysis and machine intelligence. 2012 Dec 20;35(8):1958-71.
[23] Ijjina EP. Classification of human actions using pose-based features and stacked auto encoder. Pattern Recognition Letters. 2016 Nov 1;83:268-77.
[24] Huang Z, Wan C, Probst T, Van Gool L. Deep learning on lie groups for skeleton-based action recognition. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 6099-6108).
[25] Sapiński T, Kamińska D, Pelikant A, Anbarjafari G. Emotion recognition from skeletal movements. Entropy. 2019 Jul;21(7):646.
[26] Kamencay P, Benčo M, Miždoš T, Radil R. A new method for face recognition using convolutional neural network.