Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Fakultät für Informatik und Wirtschaftsinformatik

Master-Thesis

Name:	Lukas Theurer
Thema:	Human Action Recognition in Context
Arbeitsplatz:	Inferics GmbH, Karlsruhe
Referent:	Prof. Dr.-Ing. Laubenheimer
Korreferent:	Prof. Dr. Link
Abgabetermin:	30.09.2020

Karlsruhe, 01.04.2020

Der Vorsitzende des Prüfungsausschusses

[illegible]

Prof. Dr. Heiko Körner

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Aus den benutzten Quellen direkte oder indirekt übernommene Gedanken habe ich als solche kenntlich gemacht. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Diese Arbeit wurde bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

[illegible]

Karlsruhe, September 30, 2020 Lukas Theurer

Kurzfassung

Videobasierte Analysesysteme sollen intelligenten Assistenten helfen ein gutes Szeneverständnis aufzubauen und stellen somit eine Kernkomponente moderner Assistenzsysteme dar. Als Grundlage dafür dienen oft tiefe neuronale Netze die menschlichen Handlungen auf Basis von zuvor detektierten Skeletten erkennen. Um diese Technik einem breiten Publikum zugänglich zu machen und bei diesem eine hohe Akzeptanz zu schaffen ist es essenziell die eingesetzten Netze gut verständlich und ressourceneffizient zu entwickeln. Bestehende Ansätze liefern zwar unter Laborbedingungen gute Erkennungsraten, bieten aber keine Lösungen für in der Praxis übliche Gegebenheiten wie fehlende Daten oder eine geringere Rechenleistung. Sie setzen dabei allein auf das menschliche Skelett ohne weiteres Kontextwissen miteinzubeziehen.

Eine speziell für diesen Anwendungsfall entwickelte Recognition Pipeline schafft Abhilfe: Performante Objekt Detektoren und Posenschätzer liefern erklärbare Daten die durch ein aus dem NLP-Bereich stammenden Ansatz der Wide Convolutions verarbeitet werden. Dabei werden speziell die Auswirkungen unterschiedlicher Datenvorverarbeitungsschritte sowie die Auswirkung unterschiedlichen Architekturentscheidungen beleuchtet. Die neue Herangehensweise bleibt erklärbar und gleichzeitig besonders datensparsam. Zusätzlich wird untersucht ob Kontextwissen in Form von erkannten Objekten die Klassifikationsleistung bei schwer differenzierbaren Aktionen verbessern kann.

Die vorgestellte Datenmodellierungstechniken sowie eine angepasste Netzarchitektur ermöglichen den Umgang mit fehlenden Daten und sorgen für einen schnellen Trainingsprozess. Auch wenn die aktuellen State-of-the-Art Leistungen nicht erreicht wurden, so schafft diese Arbeit eine gute Grundlage für eine Weiterentwicklung der vorgestellten Techniken.

Abstract

Video based analysis can deliver very rich scene and process information to let intelligent, electronic assistants understand the observed situation and is therefore a core component of modern assistant systems. Such systems can build on Deep Artificial Neural Networks (D-ANN), which recognize human activities by analyzing time sequences of the 3D coordinates of previously recognized body keypoints such as skeleton joints, eyes and ears. Pre-requisites to spread this technology widely and to ensure acceptance among users are the explainabilty and the ressource-efficiency of the deployed D-ANN. Existing such approaches from actual research reach already quite good recognition performance, but only under laboratory conditions. They do not offer solutions for problems encountered in real world applications, such as missing data or low available computing power on-site. On the other hand, almost all of them consider only human keypoint data without using further available context information.

These drawbacks are resolved by the context-object aware activity recognition pipeline proposed and developed in this work: Object detectors and human pose estimators deliver explainable data, which are processed by a Deep Convolutional Neural Network (d-CNN) with so-called „wide convolutions“, an approach, which was borrowed from Natural Language Processing. This work investigates the proposed solution concept especially with respect to different possible d-CNN architectures and different data pre-processing opportunities. The data processing is explainable by virtue of the chosen new approach, which is also particularly economical in terms of required sample data. It is further investigated, if context knowledge in form of recognized objects can improve the classification performance in the case of activities, which are difficult to discriminate.

The data modelling techniques and the d-CNN architectures developed and presented here allow to treat the problem of missing data and enable a fast training process, but state-of-the-art performance could not be reached on the idealized benchmark data sets, which lets space for future improvements.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Networks
CNN	Convolutional Neural Network
CV	Cross-View
CS	Cross-Subject
D-ANN	Deep Artificial Neural Networks
DTW	Dynamic Time Warping
FLOP	Floating Point Operation
GCN	Graph Convolutional Neural Net
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
JDM	Joint Distance Maps
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
NTU	NTU RGB+D
RNN	Recurrent Neural Network
SOTA	state-of-the-art
ST-GCN	Spatio-Temporal Graph Convolutional Neural Net

1 Introduction

1.1 Motivation

Intelligent assistants gained increasing societal relevance in the past years. A survey from Strategy Analytics shows an increase in sales of smart speakers by 70% in 2019 in comparison to the previous year [Strategy Analytics, 2020]. While voice-based assistants like *Alexa*[1] (Amazon) or *Siri*[2](Apple) have already found their way into the everyday life of many people, and a new generation of intelligent systems is in the pipeline: vision-based systems.

Vison-based assistants enable new opportunities in many domains, like for example the automotive sector, but also in the smart home sector. They are built to support and simplify the day-to-day processes of users and to protect the users as well. An example of such a product is the *Patronusens* sensor (see figure 1.1) which recognizes when people fall and and remain in a helpless situation.
A fundamental scene understanding by such an intelligent system is necessary in order to guarantee sufficient quality of the services. The better the scene understanding, i.e. the higher the information quality about the current situation is, the better are the decisions being made by the assistants. Important components for understanding a situation could be the arrangement of objects or the knowledge about how the humans act in the scene. The process of classifying such human actions is called *Human Action Recognition.* The Human Action Recognition can be extended with the recognition of interactions between a subject (human) and objects (humans or objects) and is called Interaction Recognition then. Today, deep neural networks are usually used for this task. While these networks are improved more and more in terms of accuracy, the complexity rises at the same time. The consequence is increasing required computing power and thus also more expensive hardware.

The high hardware cost is still an obstacle to provide these powerful algorithms to a wide public. The previous arguments and the desire for not processing private data in public clouds lead to a path called edge computing. Edge computing is the approach to process data where it is generated. For this purpose, more powerful micro controllers are used, which

[1]`https://alexa.amazon.de/` - Retrieved on: 24.09.20
[2]`https://www.apple.com/de/siri/` - Retrieved on: 24.09.20

require less power, but at the same time, have sufficient computing power to process data using deep neural networks. The *Hemistereo* sensor developed by *3DVisionLabs*[3] and shipped with Inferics software is such a device (see figure 1.1). Gartner predicts that by 2025, 75% percent of all data will be processed outside a traditional data center [Gartner, 2018]. Also, the upcoming expansion of 5G networks could have a significant impact on the applications of edge computing. Therefore, Edge AI is one of the next big topics that needs to be focused by AI research.

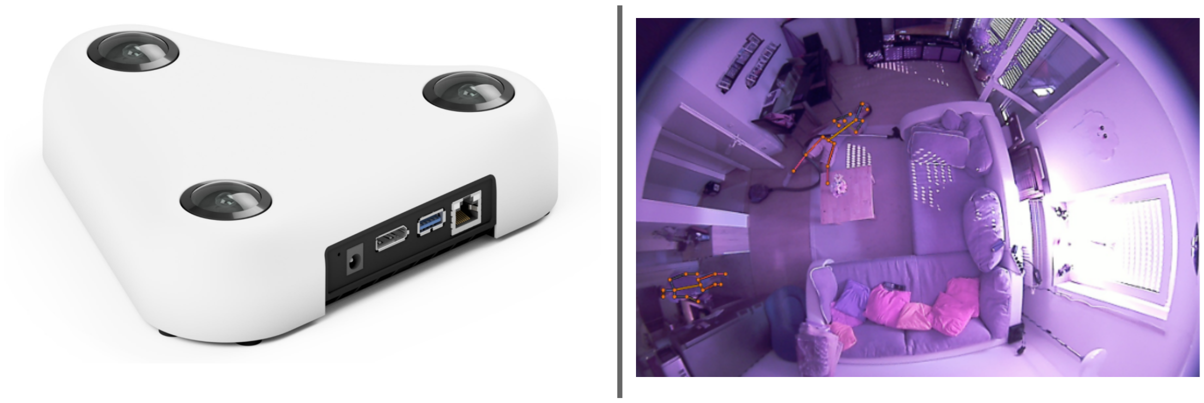


Figure 1.1: Left: HemiStereo Sensor developed by the cooperation partner 3DVisionLabs is shipped with NVIDIAs AI embedded processor Jetson TX2. The sensor is mounted on the ceiling and is able to process all data on the device itself. Right: An image captured by the sensor and processed by the inferics pose estimator.

But one must be aware that the software has to be adapted to the edge device conditions. While most state-of-the-art (SOTA) approaches are using so-called end-to-end network architectures, it is desirable to split the whole process into several smaller parts due to several reasons. First of all, splitting leads to better modularity, which makes troubleshooting and improving individual work steps easier. Secondly, it enables a better understanding of the individual processes. The latter improves the explainability of the AI and therefore leads to better user acceptance of such intelligent systems [Glikson and Woolley, 2020].

Bitkom and the German Research Center For Artificial Intelligence (DFKI) published a survey that summarizes important key aspects of upcoming AI applications [Bitkom and Deutsches Forschungszentrum für Künstliche Intelligenz, 2017]. They pointed out multiple stakeholders in the context of the interpretability of machine learning models:

[3]`https://3dvisionlabs.com/` - Retrieved on: 30.09.20

- **End-Users:** For example, a doctor gets a treatment recommendation or a mechanic who is made aware of an emerging fault case. Both are interested in the reasons for the recommendation.
- **Compliance, ethical or legal experts:** Interested in whether certain guarantees can be given for a model.
- **Data-Scientists:** Using existing models and need the chance to understand the underlying process to find issues regarding their use-case.
- **Researchers and application experts:**
- **Machine Learning Experts:** Interested in better understanding the learning algorithms themselves and not necessarily their results.

Accordingly, action recognition, modularity, and explainability play a significant role in the development of such a vision-based, electronic assistant.

1.2 Inferics GmbH

The Inferics GmbH is a startup founded in January 2018 and located in Karlsruhe. They focus on developing machine learning solutions on edge computing and embedded devices. In their latest projects, they are using a Nvidia Jetson TX2 [4] combined with 3D stereo cameras developed by their partner company 3DVisionLabs GmbH to build embedded computer vision applications (see fig 1.1). An outcome of this cooperation is their Patronusens sensor [5]. The intelligent sensor mounted on the ceiling detects a critical situation, for example a fallen person, and automatically makes an emergency call. With the work of this thesis, Inferics can improve their sensors by creating better siuation understanding by means of activity recognition. Not only in their home assistant application Patronusens but also in other surveillance use cases in which a good situation understanding is needed to support humans.

1.3 Problem Description

As already explained in the first section, a good situation understanding is essential for the decision-making process of vision-based systems. Therefore, human action recognition for adding understanding the dynamics of a situation will be investigated and implemented to classify different action classes in video sequences.

Today's state-of-the-art algorithms often use an end-to-end architecture. The advantage of these approaches is that the networks can learn a full internal representation that enables them to reach a given goal by themselves. The disadvantage is that due to the black-box nature of neural networks, these internal representations are difficult to understand. For better explanability, an action recognizer is built that integrates existing components such as networks for object detection and human pose estimation. Both are used to detect key points in a 2D RGB image. These 2D key points are then aligned with a depth image to calculate the keypoint coordinates in a three-dimensional Euklidean space. This procedure is repeated for each image in a video to get in addition to the spatial also a temporal representation. Based on this spatio-temporal data, a suitable feature representation is chosen, that is then processed by neural networks. The result is an action classifier enriched by information about detected interaction objects, which receives a video as input and delivers a corresponding action class as output.

Another downside of end-to-end approaches is that a tremendous amount of data is needed to train these deep neural networks for all combinations of scenes, persons and activities therein. This issue makes it hard to adapt a system to new environments and tasks. For example,

[4] `https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-tx2/` - Retrieved on: 30.09.20

[5] `https://www.patronusens.de/` - Retrieved on: 30.09.20

if you want to classify actions at home like vacuuming or cooking, you need to train your classification network on other data than if you want to detect some industrial use-cases. Unlike humans, current neural networks cannot transfer their knowledge to new domains. They need to be trained again on new data. To guarantee an easy training procedure with respect to the amount of data needed, the second goal of this work is to build a Human-Object-Interaction Recognition framework, which requires a smaller amount of data compared to the end-to-end counterparts.

1.4 Overview

At first, a short introduction into the field of action recognition is given. Chapter 3 introduces current state-of-the-art techniques found in a literature search. Afterwards, additional techniques for preprocessing and data handling are introduced and developed. Techniques found in the previous parts are discussed with special consideration of the new techniques. The subsequent chapter presents the implementation of the discussed techniques and the evaluation with respect to their theoretical and practical performance. In the then following discussion chapter, issues detected earlier are being evaluated and an outlook on possible improvements is given. Finally, in the last chapter the results are summarized.

2 Human Action Recognition - An Overview

2.1 Method Overview

Action recognition is a broad domain coming with different types of data and tasks. According to [Zhang et al., 2019], vision-based action recognition methods can be partitioned into the subtasks "Action Classification" and "Action Detection". While the task for classification is to classify videos, which were previously prepared (cut) to contain only one action class, in the detection task, the algorithm needs to find and classify an action in an un-processed video containing possibly multiple actions. Besides camera data, other sensor data like from gyroscopes, magnetometers, or accelerometers can be used for action recognition [Barna et al., 2019, Masum et al., 2019]. In this work, the focus lies on video-based action classification. Figure 2.1 visualizes today's action classification methods in a more granular way. The green path shows the methods used in this paper. For a detailed discussion of the listed approaches, please have a look at chapter 3.2.

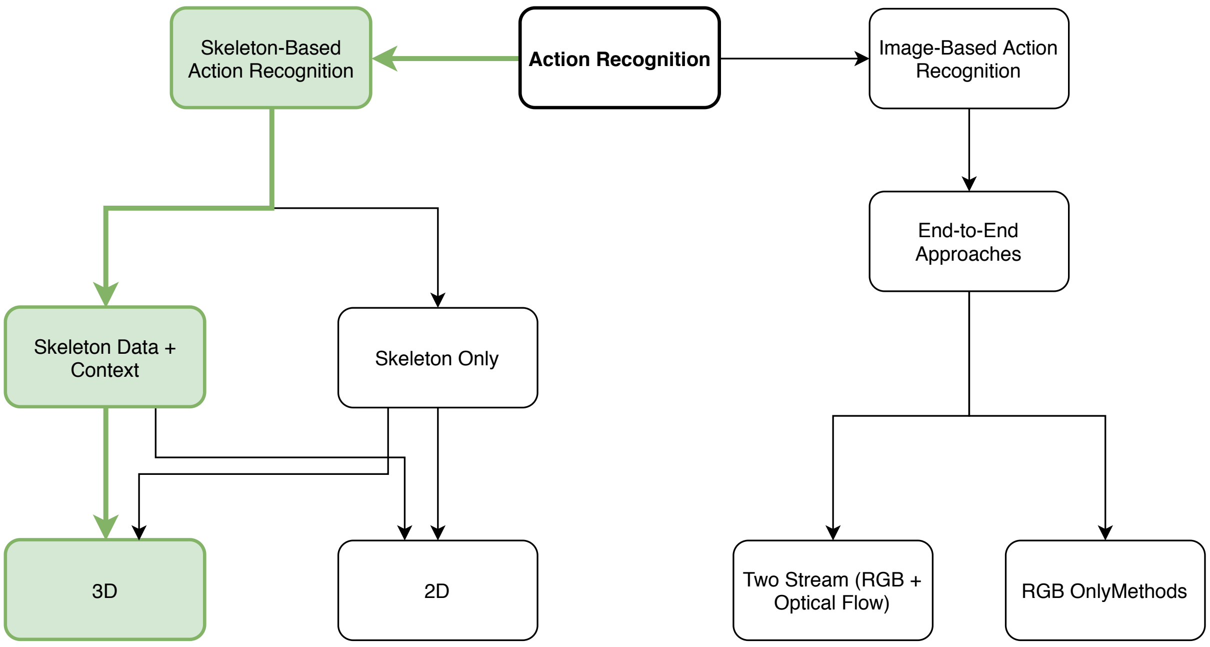

Skeleton-Based Action Recognition
Action Recognition
Image-Based Action Recognition
End-to-End Approaches
Skeleton Data + Context
Skeleton Only
3D
2D
Two Stream (RGB + Optical Flow)
RGB OnlyMethods

Figure 2.1: Overview over the proposed method and its context with other methods

The reason why the following approaches were not considered in this thesis are:

Image-Based Action Recognition: Recent benchmarks show that image-based action recognition is performing better in terms of accuracy than skeleton-based methods. However, they also require a lot of training data to model the different variances in images and their inference speed is lower. The enormous training effort, the complexity of the networks and the associated poor adaptability to new application domains are reasons against such a method.

End-to-End Approaches: Many approaches based on images are trained in an End-to-End manner. This means the networks get the ability to learn a suitable data representation themself and build upon this a robust classifier. Unfortunately, neural networks can be seen as black boxes. They take an input x, process this input, and return an output y. The process which transforms x to y is not easy to analyze and almost impossible to interpret. Processing raw data instead of meaningful features prevents the explainability of the process.

Skeleton Only: The preferred dataset used in this thesis is the NTU RGB+D Dataset [Shahroudy et al., 2016], which provides video sequences for more than 60 Action classes captured by three Microsoft Kinect V2 sensors. The dataset comes with a human skeleton annotation for each frame. Therefore most of the State-of-the-Art approaches rely only on these skeleton poses. They are often not able to get context information like image-based approaches could do. Due to the lack of this knowledge, these approaches could be improved by adding such context information.

2D: [Aubry et al., 2019] and [Avola et al., 2019], both investigated action recognition based on 2D keypoints. Although Aubry et al. were able to challenge the latest approaches in terms of accuracy, Avola did not. That does not show that 2D-Action recognition is not as good as other approaches, but requires more feature engineering. While a 3D representation enables a more view-invariant limb presentation, a 2D skeleton limb presentation strongly depends on the viewing angle of the camera. Due to the lack of information through the projective transformation, additional rich information is required to improve the classification accuracy.

2.2 NTU RGB+D Dataset

The results of this work are evaluated on the NTU RGB+D (NTU) dataset, published by [Shahroudy et al., 2016]. Besides the original version, containing 60 different action classes, an expansion, called NTU RGB+D 120 with 120 action classes, was released later. The data was captured using an arrangement of Kinect V2 sensors. [1] . The resulting video sequences contain one or two persons performing one single particular action. The correspondingly prepared videos were captured with different persons (subjects) and at different viewing angles, to represent the view variance as well as the variance of subjects separately.

Different subjects and different angles are used to benchmark new deep learning methods in two different manners:

Cross-Subject Evaluation

In Cross-Subject evaluation, the 40 subjects are split into a training a testing group of equal size. With this evaluation method, the generalization over different subjects is tested.

Cross-View Evaluation

In Cross View evaluation, all captures from camera 1 are used for testing, and cameras 2 and 3 are used for training. This way, the view-invariance of the algorithm is tested.

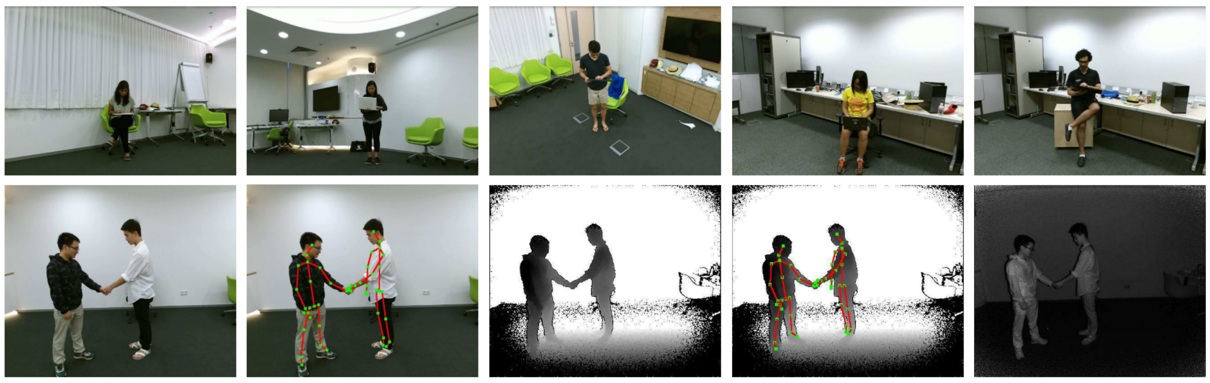


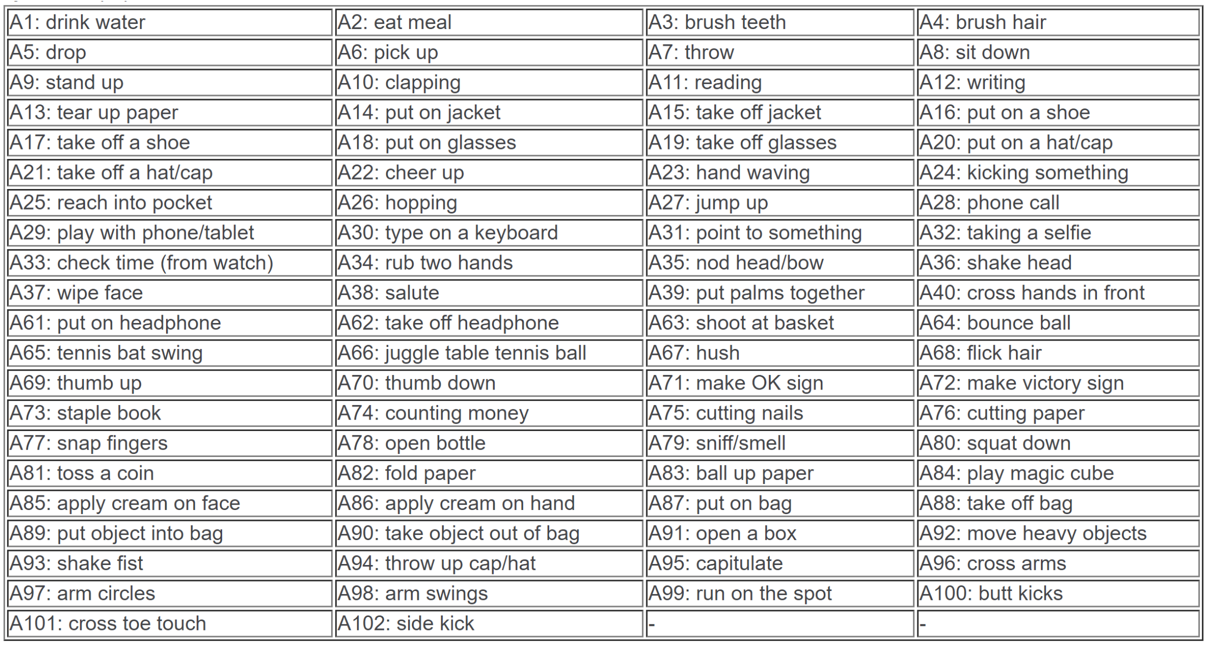
Figure 2.2: First Row: Intraclass variations by different subjects and different angles. Second Row: Example of data: RGB, Skeleton, Depth, Depth+Skeleton, Infrared

Current state-of-the-art approaches can be found on the "paperswithcode"-leaderboard [2]. The results show that in general, the generalization over different views is better than over different subjects.

[1] `https://developer.microsoft.com/de-de/windows/kinect/` - Retrieved on: 30.09.20

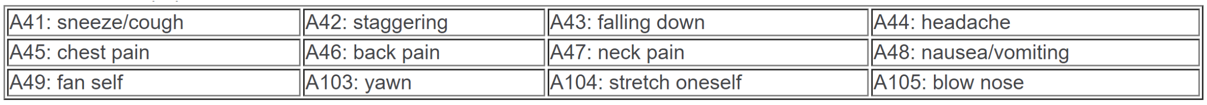
[2] `https://paperswithcode.com/sota/skeleton-based-action-recognition-on-ntu-rgbd` - Retrieved on: 30.09.20

In addition to RGB-sequences, the NTU dataset also provides 3D depth maps, human skeleton annotations, and infrared sequences. Within the 56880 video samples, we have 60 action classes performed by 40 different subjects recorded from three different views (front, left, right). The different action classes in the NTU dataset are shown in the following tables:



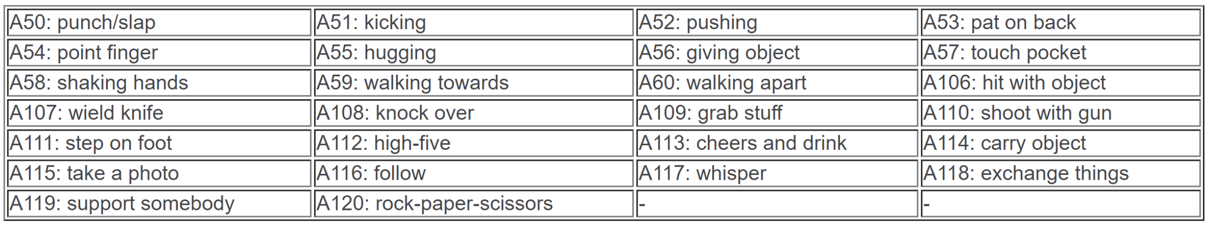
A1: drink water	A2: eat meal	A3: brush teeth	A4: brush hair
A5: drop	A6: pick up	A7: throw	A8: sit down
A9: stand up	A10: clapping	A11: reading	A12: writing
A13: tear up paper	A14: put on jacket	A15: take off jacket	A16: put on a shoe
A17: take off a shoe	A18: put on glasses	A19: take off glasses	A20: put on a hat/cap
A21: take off a hat/cap	A22: cheer up	A23: hand waving	A24: kicking something
A25: reach into pocket	A26: hopping	A27: jump up	A28: phone call
A29: play with phone/tablet	A30: type on a keyboard	A31: point to something	A32: taking a selfie
A33: check time (from watch)	A34: rub two hands	A35: nod head/bow	A36: shake head
A37: wipe face	A38: salute	A39: put palms together	A40: cross hands in front
A61: put on headphone	A62: take off headphone	A63: shoot at basket	A64: bounce ball
A65: tennis bat swing	A66: juggle table tennis ball	A67: hush	A68: flick hair
A69: thumb up	A70: thumb down	A71: make OK sign	A72: make victory sign
A73: staple book	A74: counting money	A75: cutting nails	A76: cutting paper
A77: snap fingers	A78: open bottle	A79: sniff/smell	A80: squat down
A81: toss a coin	A82: fold paper	A83: ball up paper	A84: play magic cube
A85: apply cream on face	A86: apply cream on hand	A87: put on bag	A88: take off bag
A89: put object into bag	A90: take object out of bag	A91: open a box	A92: move heavy objects
A93: shake fist	A94: throw up cap/hat	A95: capitulate	A96: cross arms
A97: arm circles	A98: arm swings	A99: run on the spot	A100: butt kicks
A101: cross toe touch	A102: side kick	-	-

Table 2.1: NTU RGB+D Daily Actions



A41: sneeze/cough	A42: staggering	A43: falling down	A44: headache
A45: chest pain	A46: back pain	A47: neck pain	A48: nausea/vomiting
A49: fan self	A103: yawn	A104: stretch oneself	A105: blow nose

Table 2.2: NTU RGB+D Medical Actions



A50: punch/slap	A51: kicking	A52: pushing	A53: pat on back
A54: point finger	A55: hugging	A56: giving object	A57: touch pocket
A58: shaking hands	A59: walking towards	A60: walking apart	A106: hit with object
A107: wield knife	A108: knock over	A109: grab stuff	A110: shoot with gun
A111: step on foot	A112: high-five	A113: cheers and drink	A114: carry object
A115: take a photo	A116: follow	A117: whisper	A118: exchange things
A119: support somebody	A120: rock-paper-scissors	-	-

Table 2.3: NTU RGB+D Interactions

2.3 Recognition Pipeline

In order to better understand the classification approaches of this work, presented in the following chapter, this section introduces the applications pipeline architecture.

In the first chapter, a multi-stage approach was mentioned, that should ensure a more natural and better understandable recognition process. [Baradel et al., 2018] developed a model that performs reasoning in a sequence of images related to human behavior. In contrast to previous works, Baradel et al. extended an end-to-end architecture by an additional head called Object Relation Network (ORN). The ORN uses information extracted from an object mask detector (Mask R-CNN [He et al., 2017]) and a second Convolutional Neural Network with two heads that extracts features of objects and features of activities. These are then processed by a Gated Recurrent Unit (GRU) [Cho et al., 2015], which enables the caption of long term relations over multiple frames. They outperformed state-of-the-art approaches and showed that the additional high-level object information leads to a performance increase.

The usage of already trained detectors as feature extractors cannot only improve the overall accuracy, as Baradel showed. It also leads to recognition system modularity where worse-performing modules can be exchanged easily with other algorithms performing the same task better. Ideally, if they provide the same output features, the dependency between the front-end pose estimator and object detector and the back end encoding and action classifier is low, which leads to even better modularity. Otherwise, if the extractor output changes, the encoding needs to be adapted and respectively, the action classifier too, because the input feature space was changed. However, in contrast to an end-to-end network architecture that needs to be adapted to every single change, the modularity allows us to reuse the components for other tasks as well. An example for such a scenario could be training different feature extractors using the same COCO dataset that provides a fixed number of key points. This way, the architecture can simply be changed while keeping same input and output shapes.

Inspired by this approach, a multi-stage architecture for Human Action Recognition is developed similar to Baradels reasoning pipeline. The usage of object detectors as feature extractors for high-level semantic and meaningful features enables new opportunities concerning training data usage and explainability. The pipeline is visualized in Fig 2.3. Due to the possible restriction that object detectors, as well as the pose estimator, operate on single images, each image is processed separately by the feature extractor of the pipeline and then stored until the a whole action sequence was processed. The results are then transformed and encoded with an encoding module. Afterwards, neural nets are processing the encoding result and predicting a particular action class for the input sequence.

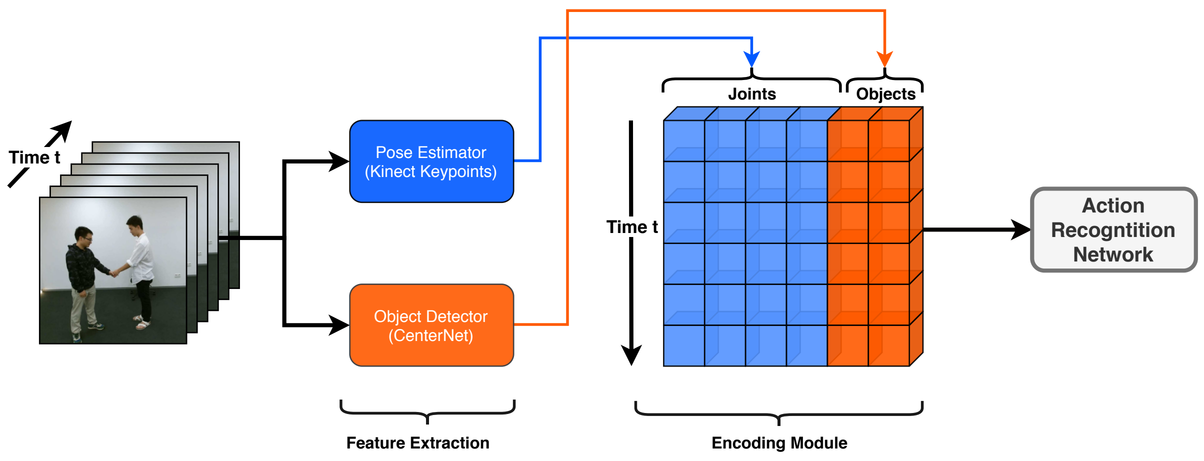

Time t
Pose Estimator (Kinect Keypoints)
Object Detector (CenterNet)
Joints
Objects
Time t
Action Recogntition Network
Feature Extraction
Encoding Module

Figure 2.3: The proposed pipeline structure.

In the following chapter 3, different classifier approaches are introduced. Besides Convolutional Neural Network (CNN) architectures, Recurrent Neural Networks (RNNs) and Graph Convolutional Neural Net (GCN) approaches for action recognition are discussed. The period of our work did not let enough time to optimize the person and object keypoint extractors (detectors). Instead, a pretrained CenterNet [Duan et al., 2019] detector is used to extract object keypoints from RGB data and the person keypoint data ("skeletons") are taken from the NTU dataset as provided. See chapter 4.3.5 "Feature Extraction" for further information.

3 Related Work of Machine Learning Methods for Skeleton-Based Action Recognition

3.1 Traditional Methods for Handling Sequential Data

Due to the lack of sufficient computational power in the past and, therefore, the missing ability to process data with deep neural networks, the first human action recognition works aimed to model the sequential data efficiently. This requires the consideration of the temporal and the spatial properties of the data. In the time domain, the sequences may vary in length, speed or in the order of action execution. Former systems had problems processing sequences if those contained unforeseen activity elements.

3.1.1 Dynamic Time Warping

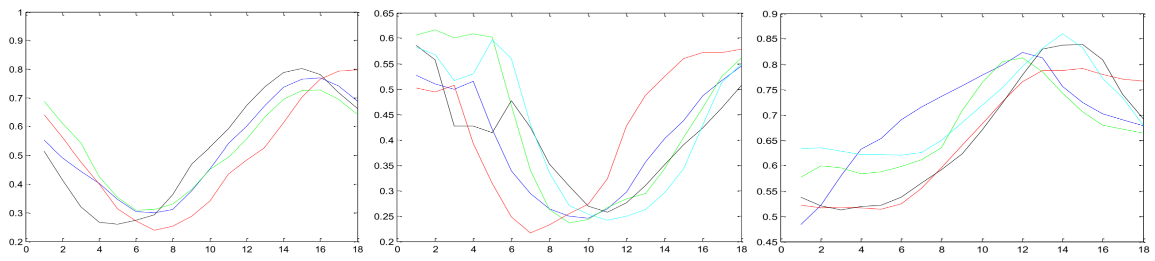
Dynamic Time Warping is an approach that maps a sequence Y to a reference sequence X containing similar data points [Senin, 2008]. For this purpose, a distance function needs to be defined. Then, the distance between each data point x_{t1} and y_{t2} is calculated and stored in a distance matrix D. Additionally a few constraints need to be defined as in [Sempena et al., 2011]:

Boundary Condition: The starting point of X and $Y : p_1 = (1, 1)$; as well as the endpoint $p_K = (N, M)$, must be the first and, respectively, the last points of the aligned sequence.

Monotonicity condition: $n_1 \leq n_2 \leq \ldots \leq n_K$ and $m_1 \leq m_2 \leq \ldots \leq m_K$. So the observated points are aligned in order of time.

Step size condition: $k_{i+1} - k_i \leq 1$ (No observation symbols may be skipped)

[Sempena et al., 2011] and [Chaaraoui et al., 2013] uses a Dynamic Time Warping (DTW) approach to classify actions in sequences. They representing a pose at time step t via a joint orientation over time series as quaternion to be size and position invariant. After the DTW, a Nearest Neighbour Classifier is used to classify actions like "clap", "punch", or "wave". [Chaaraoui et al., 2013] et al. use a DTW to build an action recognition framework based on skeletal and silhouette-based features and normalize their representation over time by applying DTW.



(a) DTW Clap (b) DTW Punch (c) DTW Smash

Figure 3.1: Different actions transformed to process them with Dynamic Time Warping by [Sempena et al., 2011]

Whereas DTWs perform sufficiently good in the early days, their disadvantages are the quadratic time and memory complexity when aligning large sequences. Although multiple enhancements made to deal with this issues, [Bagnall et al., 2016] compares multiple time series classification algorithms and pointed out 12 algorithms that significantly outperform DTW. In addition, appropriate reference points, that will be used for time-series comparison, need to be defined by the user itself. For today's deep learning systems this step can be skipped because an Artificial Neural Networks (ANN) learns the semantic rich points of interest in a sequence.

With DTW, a transforming algorithm was presented, which enables the processing of inputs of varying sizes for classifiers that usually cannot handle sequential data.

3.1.2 Hidden Markov Models

Hidden Markov Models, another time series processing technique was introduced in 1986 by [Rabiner and Juang, 1986]. It is are based on the Markov Chain Rule. "An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden) but can only be observed through another set of stochastic processes that produce the sequence of observed symbols" [Rabiner and Juang, 1986].

[Gedat et al., 2017] for example, proposed an Hidden Markov Model (HMM) in combination with an ANN to classify human actions based on poses. For this purpose, they developed a pose alphabet (see. Fig. 3.2). The ANN then classifies a set of 22 body parts as one of the poses of the alphabet. Afterwards, the neural net output is processed by multiple HMMs to classify a sequence of observed poses according to the classes: the right step, the left step, and the right stroke.

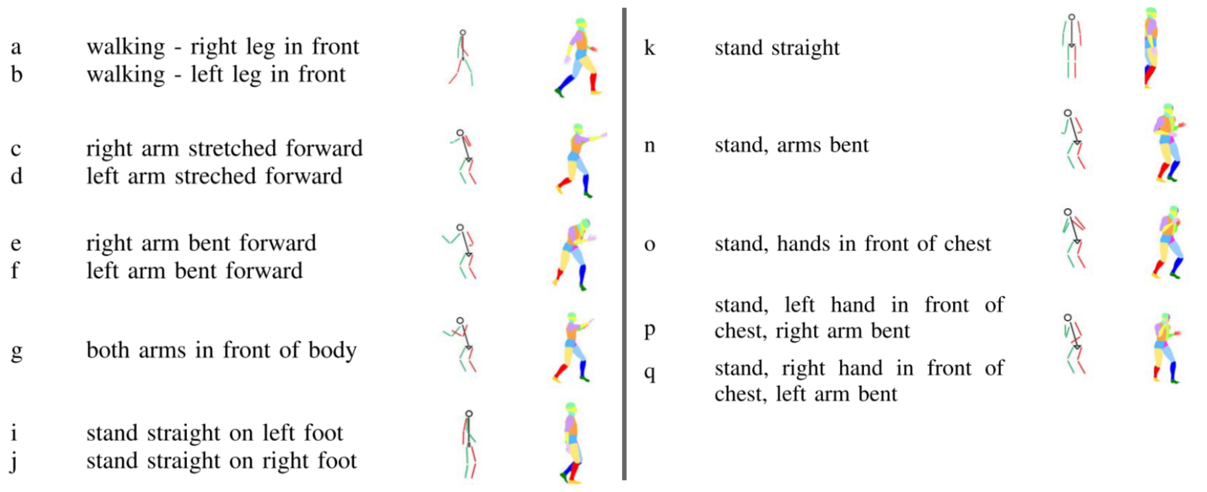

a walking - right leg in front
b walking - left leg in front
c right arm stretched forward
d left arm streched forward
e right arm bent forward
f left arm bent forward
g both arms in front of body
i stand straight on left foot
j stand straight on right foot
k stand straight
n stand, arms bent
o stand, hands in front of chest
p stand, left hand in front of chest, right arm bent
q stand, right hand in front of chest, left arm bent

Figure 3.2: Proposed pose alphabet [Gedat et al., 2017]. Single action primitives are extracted by an ANN and further processed by a Hidden Markov Model

[Chakraborty and Talukdar, 2016] stated the following issues and limitations of HMMs in speech processing. Due to its close relationship to the human action recognition topic, a few points can be adapted:

1. **The Markov assumption:** The Markov property says that being in a state S_t at time t does only depend on the previous state S_{t-1}. The following example shows that this does not apply to actions: If we want to predict whether a person is asleep or awake based on movement data. The chance of someone transitioning from asleep to awake increases, the longer the person has been in the sleep state. So this shows that not only the previous state but also multiple previous states can be crucial [1].
2. **Successive observations are independent:** Other than ruled by the Markov assumption, "[...] successive observations, in reality, are rarely independent of each other." [Chakraborty and Talukdar, 2016]
3. **Inconsistencies in data:** Variation like missing data or data recorded by different sensors need to be normalized in order to use HMMs
4. **The number of states needs to be predefined:** Because the states are the discriminant factor in HMMs, the number of used states is crucial for performance. A developer does not only needs to define which data points are representative for a particular action. He also needs to define how much of these states are required.

While Assumption 1 and 2 show that the Markov Property does not match the action recognition use case, point 3 pointed out that data modeling is a crucial aspect in Hidden Markov Models.

[1] `https://stats.stackexchange.com/a/366430` - Retrieved on: 20.09.20

Already described in point 4, the number of states that need to be selected possibly leads to a non-optimal setting. Actions could be too complex to break them down onto a few key frames. Newer approaches can handle the last two problems better. This will be discussed later.

3.1.3 Conclusion

With Dynamic Time Warping and Hidden Markov Model, two different approaches were shown, which can handle sequential data. Both were developed in the 20th century and have therefore been used for a long time successfully. However, also their disadvantages were shown: Dynamic Time Warping is just a preprocessing process that needs a preprocessing themselves and a definition of a distance metric for matching sequence parts. Same for Hidden-Markov Models. They can perform well if the parametrization is perfectly balanced, but this requires more Trial-and-Error engineering than in recent approaches.

Due to the amount of data available today and the enormous computing capacity while training, deep learning methods in particular benefit from nowadays conditions. These would also benefit from sophisticated feature engineering, but are not as dependent on it as the two methods presented in this chapter.

3.2 Deep Learning for Action Recognition

With Dynamic Time Warping and Hidden Markov Model, two different approaches were shown, which can handle sequential data. They can perform well if the parametrization is perfectly balanced, but this requires more Trial-and-Error engineering than in recent approaches. This issue leads to usage of nowadays deep neural networks that. In this chapter, CNNs, GCNs and RNNs approaches are shown, advantages and disadvantages are discussed, and techniques that could improve these concepts are introduced.

3.2.1 Convolutional Neural Networks

Convolutional Neural Network (CNN) became very popular in the computer-vision society after [Krizhevsky et al., 2012] reached superior results on the ImageNet dataset[2] outperforming all previous approaches. From this, interest in deep learning increases more and more. Due to the recent successes in image processing, CNNs were also researched in other domains like Action Recognition, Natural Language Processing (NLP), and signal processing in general. In this chapter, multiple approaches of Convolutional Neural Networks in action recognition are investigated.

[2]`http://www.image-net.org/` - Retrieved on: 30.09.20

3.2.1.1 Action Recognition with Joint-Distance Maps

Most of the current action classification approaches using conventional CNNs are aiming at the representation of skeleton-based data into the form of RGB image data.

[Li et al., 2017a] study an encoding technique based on the distances between skeleton joints. They called them Joint Distance Maps (JDM). Their research is based on two known issues: First, the earlier in [Wang et al., 2016] proposed mapping of the cartesian joint coordinates (x,y, and z) into a three-dimensional RGB like space, where the authors "[…] encode joint trajectories into texture images and capture temporal information by mapping the trajectories into a hue, saturation, value space"[Li et al., 2017a]. However, the problem of their method is that it is not view-point invariant. The second was that they determine that recurrent neural nets "tend to overemphasize the temporal information" [Li et al., 2017a]. They concluded that Convolutional Neural Networks are better for recognizing actions, but the existing encoding techniques are not sufficient enough.

Li expressed the joint distances in the context of interaction instances $G = \left\{p^1, p^2, \ldots, p^t\right\}$ with joints p in frames m.

$$p_j = (px, py, pz), j \in \{1, \ldots, m\}$$

The distances are then calculated as sequence of their temporal order $H_{jk} = \left\{D^1_{jk}, D^2_{jk}, \ldots, D^t_{jk}\right\}$ where D is the Euclidean distance denoted as $D^i_{jk} = \left\|p^i_j - p^i_k\right\|_2, \quad j, k \in m; j \neq k$. Therefore the columns hold the spatial representation and the rows a temporal representation. This step is followed by a bilinear interpolation upsampling from t to t'.

Instead of using absolute joint coordinates, Li uses difference vectors to encode the joint coordinates. To overcome the issue of different persons having different sizes, an additional normalization needs to be applied. Instead of normalizing the distances on the skeletal level [Zanfir et al., 2013], the normalization is applied at the transformation step. This is done by mapping the joint distances between zero to the maximum value to the color map value range (0-255).

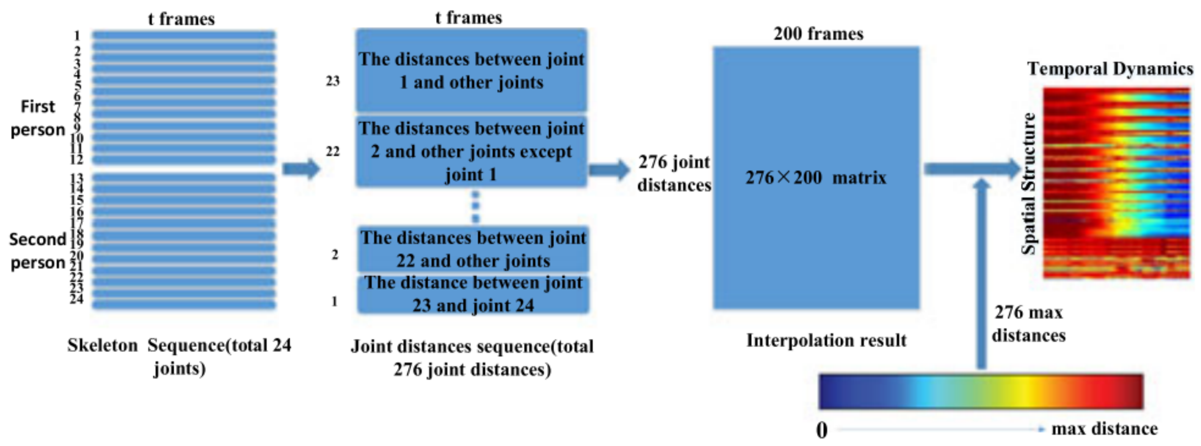

t frames
First person
Second person
Skeleton Sequence(total 24 joints)
t frames
The distances between joint 1 and other joints
The distances between joint 2 and other joints except joint 1
The distances between joint 22 and other joints
The distance between joint 23 and joint 24
Joint distances sequence(total 276 joint distances)
276 joint distances
200 frames
276×200 matrix
Interpolation result
Temporal Dynamics
Spatial Structure
276 max distances
0
max distance

Figure 3.3: Process of mapping joint distances to colors proposed by [Li et al., 2017a]

The Joint Distance Maps are calculated for distances in the xy plane, the xz plane, the yz plane and with all three dimension xyz. For recognition, four of these JDMs are built (xy, xz, yz, xyz). In Fig 3.4 , the proposed network architecture is shown. The four JDMs are input into four separated CNNs. Each outputs a Softmax classification score, which is fused by an element-wise multiplication. This fused score represents the final class probability.

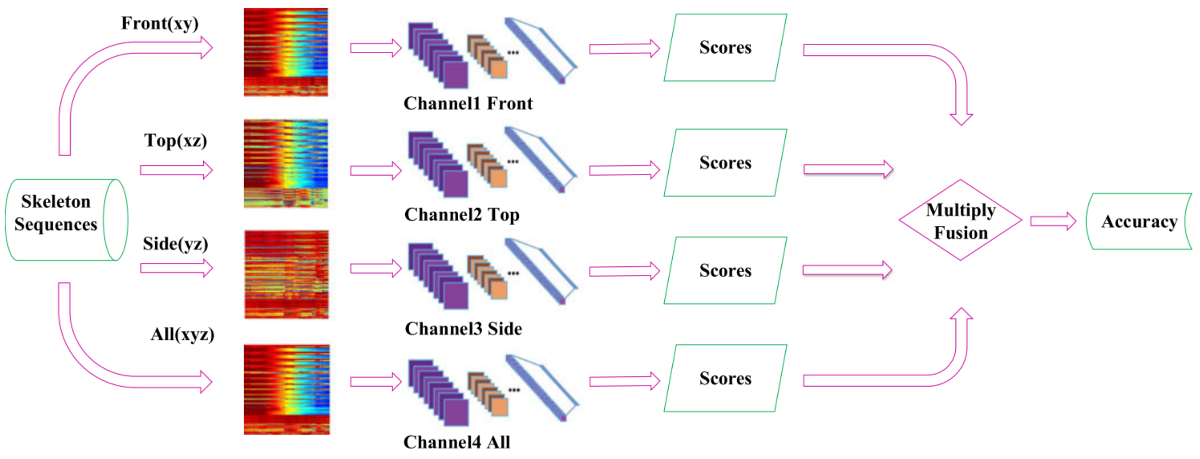


Figure 3.4: The by [Li et al., 2017a] proposed network architecture

However, Li et al. tested their method on the NTU RGB+D (NTU) dataset as well as on the UTD-MHAD [Chen et al., 2015] Dataset. On NTU dataset, they reached a Cross-subject accuracy of 76.2% and a Cross-view accuracy of 82.3%. With this result, they outperform most of their competitors. Especially, Li et al. compared the results with recurrent net approaches. Investigations showed that recurrent nets can not coping long-term dependencies in sequences.

Although Long Short-Term Memorys (LSTMs) [Hochreiter and Schmidhuber, 1997] are treating long-term dependencies, they "tend to overemphasize the temporal information especially when the training data are not sufficient, leading to overfitting" [Li et al., 2017a].

3.2.1.2 Convolutions on raw skeletal data

Another CNN-based approach was introduced in 2017 by [Li et al., 2017b]. Instead of transforming the key points into a color space, Li used 3D and motion information directly without normalizing them.

Li et al. adapt the Faster R-CNN model [Ren et al., 2015] mostly used in image classification and object detection to the domain of skeleton-based action recognition. The Faster R-CNN can be divided into two stages: The first stage is a so-called “Region Proposal Network”, which guesses the position of objects in an image, and the second stage refines these proposals and classifies it. To use this architecture, Li exchanged the 2D convolution kernels by 3D kernels, to convolve also in the temporal dimension. In addition, the region proposal part, which was originally defined to find regions of interest in an image, was refined to find informative frames in an image sequence. This region proposal part was mainly important for the action detection task, wheras this thesis only perform action classification, here also referred as action recognition.

Two-Stream CNN

According to other successful algorithms, [Li et al., 2017b] used a dual-stream approach, where two different data types are processed by two different parts of the neural net. The first stream is the raw skeleton data S, the second one the motion information of two consecutive frames M. Raw skeleton joints consisting of x,y, and z coordinates are stacked. Each skeleton S consists of N joints J and $J = (x, y, z)$. The motion information for the second stream is simply the difference vector of a joint in two consecutive frames: $M = S^{t+1} - S^t = \left\{ \boldsymbol{J}_1^{t+1} - \boldsymbol{J}_1^t, \boldsymbol{J}_2^{t+1} - \boldsymbol{J}_2^t, \dots, \boldsymbol{J}_N^{t+1} - \boldsymbol{J}_N^t \right\}$ where t is the frame index.

Skeleton Transformer

While in image processing, the order of data is crucial for the detection with CNNs, the order of the joints of a skeleton is arbitrarily chosen, which might not be optimal for action recognition. Hence, Li et al. applied a skeleton transformer, which learns a sufficient order by performing a linear transformation $\boldsymbol{S}' = \left(\boldsymbol{S}^T \cdot \boldsymbol{W}\right)^T$, where $\boldsymbol{W}$ is an $N \times M$ weight matrix. This mechanism enables the network to select important joints itself and can be seen as a simple version of attention mechanism. It can be implemented by a simple, fully connected layer without bias. Li et al. added this layers before the first CNN layers so that the network can be trained in an end-to-end manner.

Comparison of different data types

Li compared in an ablation study the performance of their method using different data combinations as input on NTU RGB+D dataset (table 3.1). They pointed out that the CNN with motion information leads to a greater improvement in term of Cross-View (CV) and Cross-Subject (CS) accuracy (81.4% and 88.5%) than the CNN and the transformer module (81.6% and 85.4%). A combination of both advanced techniques improves the overall accuracy and outperforms former SOTA approaches.

Method	Cross-Subject	Cross-View
CNN	79.8	85.2
CNN+Motion	81.4	88.5
CNN+Transformer	81.6	85.4
CNN+Motion+Transformer	**83.2**	**89.3**

Table 3.1: Ablation study on action classification. Accuracies are measured on validation set of the NTU RGB+D dataset

3.2.1.3 Conclusion

[Li et al., 2017b] and [Li et al., 2017d] introduced techniques that enable the processing of skeleton-based data with classical CNNs. Both showed that CNNs are applicable to sequential data and can even compete with recurrent neural networks.

3.2.2 Graph Convolutional Networks

In addition to classical convolution networks and recurrent neural networks, graph convolution networks in particular currently achieve the best results in the field of skeleton-based action recognition [Kim et al., 2019, Shi et al., 2019]. The idea of Graph Convolutional Neural Nets is to enable the processing of data lying in a non-Euclidean space like, for example, social networks. Early, they were adapted for human action recognition, where they represent the human skeleton as a graph. 2018, [Yan et al., 2018] introduced the Spatio-Temporal Graph Convolutional Neural Net (ST-GCN) where the joints are the vertexes, and the limbs are the edges of the graph. Besides this spatial representation, a temporal connection is made by connecting the joints with their consecutive representatives. Derived from Yan et al., two state-of-the-art approaches were introduced. Both criticized the human body-based graph structure used by Yan. “The skeleton graph used in ST-GCN is heuristically predefined based on the natural connectivity of the human body. Thus it is not guaranteed to be optimal for the action recognition task” [Shi et al., 2019]. Therefore, the two SOTA approaches try to tackle this problem.

3.2.2.1 New Sufficient Features for Action Recognition

In 2019, Kim et al. developed an action recognition algorithm based on the previous introduced ST-GCN that tackles multiple problems [Kim et al., 2019]:

1. Video sequences contain consecutive frames holding redundant information
2. The human body based graph might not be sufficient for action recognition
3. Human body information could be enriched by adding object-related information

In the following, Kims solution for these three arguments is introduced.

Video sequences contain consecutive frames holding redundant information:
Kim et al. proposed a technique called "informative frame selection strategy." They address a problem not mentioned by [Shi et al., 2019] or [Yan et al., 2018]. "The conventional skeleton-based action recognition methods assume that accurate human skeletons are provided by datasets. Real-world scenarios, however, do not provide skeletal data, and the estimated skeletal data might be inaccurate due to an unusual appearance or missing or overlapping body parts" [Kim et al., 2019].

To select important frames, first, the entire video sequence is split into T segments with equal interval $\{S_1, S_2, \ldots, S_T\}$. Each sequence consists of M frames. Then, for each frame, a pose estimation net OpenPose [Cao et al., 2016] is used to find occurring human bodies. OpenPose also determines a confidence score for each joint, which is summed up to calculate an overall confidence for one frame. Finally, the frame with the highest confidence sum is used as a representative sample for a segment S_T. These selected frames are used to model the skeletal graph. Kim et al. showed that their informative frame sampling strategy leverages the activity recognition process. Comparing with ST-GCN, they improve the accuracy on their own created Illegal Rubber Dumping dataset (IRD) by 5.5% and on ICVL-4 [Jin et al., 2017] by 8.8%.

The human-body based graph is not sufficient for action recognition, and human body information could be enriched by adding object-related information:

As already mentioned in the introduction of this chapter, the human body-based graph might be inefficient for action recognition. Due to this, Kim et al. investigated multiple hand-crafted body representations and compared them to the by ST-GCN used graph representation. The following subsets were created:

- **Object-Human-Pose (OHP)-body stream:** A graph constructed by connecting all human joints with their real neighbors and the involved object.
- **OHP-limbs stream:** Only the limb involved joints are connected with each other and the object
- **OHP-hands stream:** Both hands are connected with objects
- **HP stream:** No connection between the human body and the object. Only the human skeleton is considered

Combined with the following sampling strategies:

- **Uniform samples:** Normal sampling strategy. Every frame is used for information extraction.
- **Informative samples:** The sampling strategy described in chapter 3.2.2.1 is used.

Kim et al. have shown that switching from HP stream to OHP-hands stream alone does not lead to any improvements (See table 3.2), but using the informative sampling strategy in combination with OHP-Hands could leverage the action recognition process. This led them to the conclusion that the "frame selection strategy is more informative to the human pose graph than the object-related graph" [Kim et al., 2019]. By Combining the human pose graph, the OHP-hands stream, and the informative sampling strategy in a multi-stream network approach, they achieved the best results. With their research, Kim et al. demonstrated that building a relation between every joint and the object does not lead to any improvement. Probably because of unnecessary data. But they have also shown that the human-body pose graph profits from additional object-hand information.

Method	IRD (%)	ICVL-4 (%)
ST-GCN [39] (HP stream + uniform samples)	74.03	80.23
OHA-GCN (OHP-body stream + informative samples)	71.27	72.09
OHA-GCN (OHP-limbs stream + informative samples)	73.48	76.74
OHA-GCN (OHP-hands stream + uniform samples)	74.59	79.06
OHA-GCN (OHP-hands stream + informative samples)	76.24	81.40
OHA-GCN (HP stream + informative samples)	79.56	88.37
OHA-GCN (Two stream; HP + OHP-hands + informative samples)	**80.11**	**91.86**

Table 3.2: HP: Human Pose, OHP: Object-related Human Pose | Comparison of image-based and skeleton-based models

Besides the previously introduced innovations, Kim et al. compared in their ablation study skeleton-based action recognition with image-based action recognition. The latter is done by applying CNNs. Kims studies have shown, that in terms of accuracy, Convolutional Neural Network are better compared to their models. On the one hand, accuracy is the metric with which newly developed methods measure themselves against the old ones. However, on the other hand, it is not the only important metric for real-world applications. An appropriate inference speed is necessary to bring deep learning to edge devices. Kim has shown, that skeleton-based approaches are up to 25-times faster than image-based action recognition approaches and reach a similar accuracy (See table 3.3). However, it should not be ignored that for skeleton-based action recognition, pose estimators are necessary, which also require time for inference.

Method	Acc.(%)	Speed (ms)
CNN (LeNet [21])	87.21	3.66
CNN (VGG16)	95.35	24.56
CNN (ResNet50)	**96.51**	17.26
ST-GCN	80.23	0.1321
OHA-GCN (HP stream)	88.37	**0.1306**
OHA-GCN (OHP stream)	81.40	0.1319
OHA-GCN (Two Stream)	91.86	0.2447

Table 3.3: Comparison of different pose encoding strategies in term of accuracy and speed

3.2.2.2 Learning connections improve skeleton-based action recognition

In [Shi et al., 2019], the ST-GCN is also taken as a baseline but also criticized with respect to the fixed human body model on which the graph is built. They name the following problems with ST-GCN :

1. "The skeleton graph used in ST-GCN is heuristically predefined based on the natural connectivity of the human body. Thus it is not guaranteed to be optimal for the action recognition task. For example, the relationship between the two hands is important for recognizing classes such as “clapping” and “reading.” However, it is difficult for ST-GCN to capture the dependency between the two hands since they are located far away from each other in the predefined human-body-based graphs."
2. "The neural networks are hierarchical, where different layers contain different levels of semantics. However, the topology of the graph applied in ST-GCN is fixed over all the layers, which lacks the flexibility and capacity to model the multi-level semantics contained in different layers."

3. "One fixed graph structure may not be optimal for all the samples of different action classes. For classes such as “wiping face” and “touching head,” the connection between the hands and head should be more reliable, but it is not valid for some other classes, such as “jumping up” and “sitting down”."

Shi suggests learning the connection between the joints adaptively, depending on the action class. For this purpose, a multi-stream network architecture is built (Fig 3.5). They pointed out that first-order information (coordinates of the joints), as well as the second-order information (direction and length of the bones), are “worth to be investigated for skeleton-based action recognition”[Shi et al., 2019]. In addition to the spatial connections of the graph, Shi et al. connect the joints and limbs in temporal dimension with their consecutive neighbors at time $t + 1$ (see green connections in Fig 3.7). In the following subsection, different aspects of Shis graph convolution approach will be introduced and discussed.

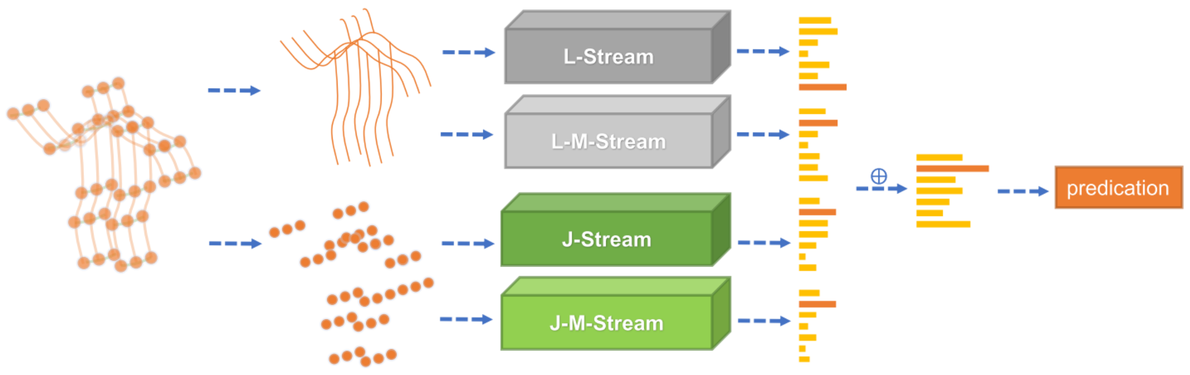


Figure 3.5: The in [Shi et al., 2019] proposed multi-stream attention-enhanced adaptive graph convolutional neural network. The output of each stream is a Softmax vector. All four vectors are fused by a simple element wise multiplication to get a final action score.

Multiple Graphs for better action recognition

The graphs shown in figure 3.5 on the left are again divided into two subgraphs. The first subgraph B_k is the global graph without any restriction in spatial connection. The global graph is used to learn the dependency of two vertexes for each action independently. The elements of B_k are learned and updated in the learning process, so there are no constraints. B_k is different for each layer, displayed in figure 3.6. It is initialized by the well-known human body model (figure 3.7 left).

The second subgraph should learn the strength of the link between two vertexes. To measure the similarity between two vertexes, two embedding functions, θ and ϕ are learned to transform feature maps into an embedding space. These embeddings are then processed by a softmax-like function and multiplied by each other. The result is a matrix where each position holds a score that represents the importance of the edge between two vertexes for a particular action.

The global graph learns and detects the graph topology for action recognition in general, while the individual graph "adds individuality according to the various sample features" [Shi et al., 2019]. A gating mechanism then fuses the information of both subgraphs.

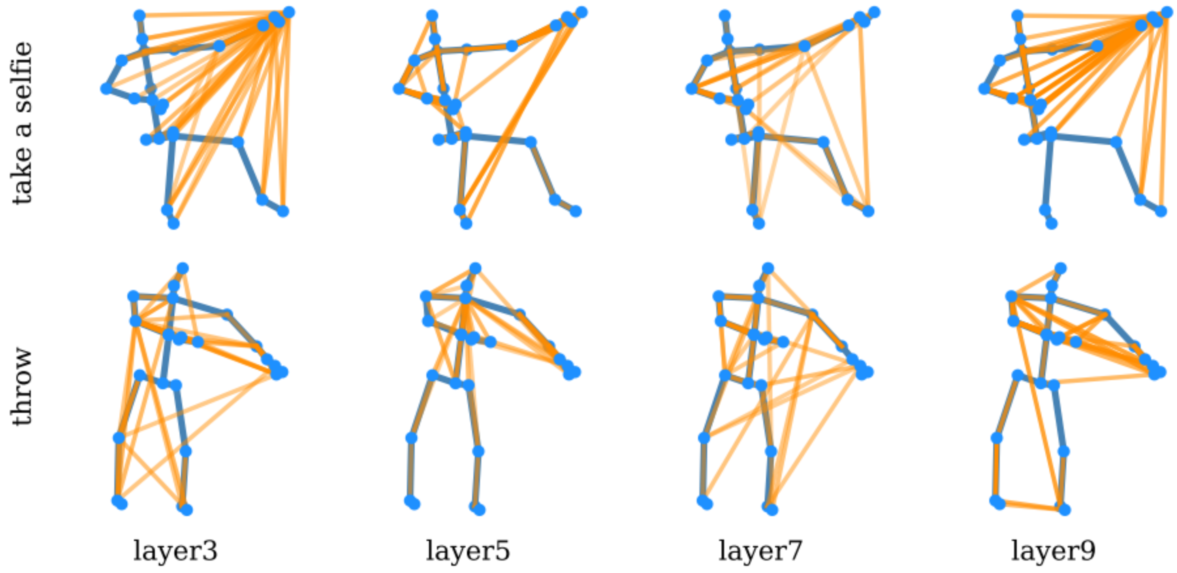


Figure 3.6: Examples of the learned graph topologies. The orange lines represent the connections. [Shi et al., 2019]

Prepare skeleton-based graph for convolutions

To apply convolutions to a graph, a few adaptions of the classical convolutions are necessary. While applying the kernel of convolution to images is easy because of its matrix structure, a vertex in a graph can have a different number of neighbors, on which the convolution is applied. Due to that reason, Shi developed a process where the convolutions are applied to sampling areas of a skeleton graph. Instead of convolving over the whole skeleton, a sampling area B_i is built with the help of a "center of gravity" (**X** in fig 3.7). The right side of the same figure shows the sampling strategy:

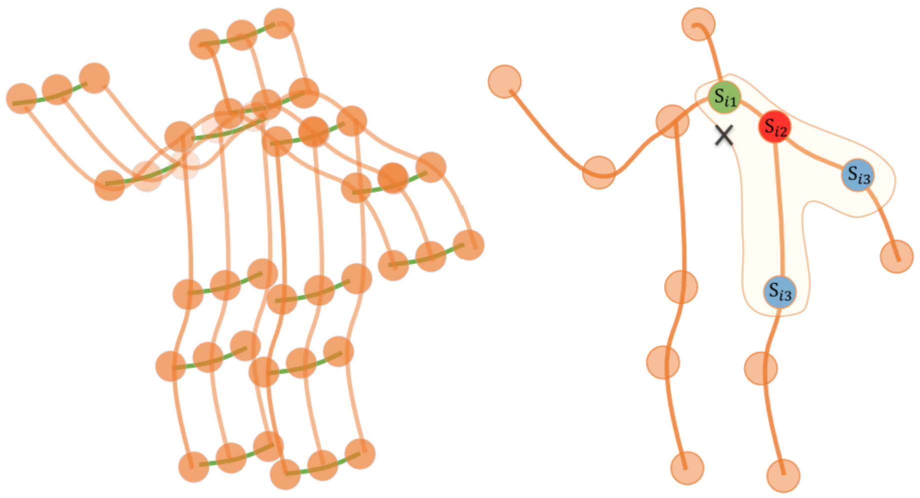


Figure 3.7: Left: Illustration of the human body based graph. In addition to the spatial connections (orange), the vertexes are connected to their consecutive neighbors in the next frame (green). Right: Sampling strategy used to prepare data for graph convolutions.

A vertex S_{i2} is chosen as “target vertex” v_i. Then, the 1-neighbor vertexes that are closer to the center of gravity are chosen as a centripetal subset ($S_{i1} green$), and the 1-neighbor vertexes, which are further away than v_i are chosen as a centrifugal subset (S_{i3} blue). The third subset is the vertex v_i itself.

The convolution equation is the following:

$$f_{out}\left(v_i\right) = \sum_{v_j \in \mathcal{B}_i} \frac{1}{Z_{ij}} f_{in}\left(v_j\right) \cdot w\left(l_i\left(v_j\right)\right)$$

B_i is the sampling area with vertexes v. w is the weighting matrix similar to the classic convolution which provides a weight vector. While the size of the weight vector is fixed, the number of nodes contained in subset B_i is not. Therefore, a mapping function l_i is necessary, to map the input to the same dimension as the convolution. $Z_i j$ denotes the cardinality of subset S_{ik} that contains v_j.

3.2.2.3 Conclusion

The by [Shi et al., 2019] proposed algorithms for skeleton-based action recognition with graph convolutions in videos are not sufficient for real-world application because of the following reasons:

1. The graph can only be constructed correctly if no joints are missing. If the connection to one of the limbs is broken, the resulting information loss cannot be recovered.
2. According to Shi's suggestion, the convolution is performed on the three subsets of the nodes of the graph. The construction of these subsets leads to a lack of information by merging nodes.
3. An important "center of gravity" needs to be calculated at each step. If joints are missing, the position of the center of gravity could have a high variance, and the subset formation result in different outcomes.

Shi's approach to model connections between joints individually might be useful for skeleton-based action recognition in videos. Nevertheless, initializing the graph according to the human skeleton, could lead to an unintended weighting of nodes. Therefore, a random initialization of weights could improve the learning process and better action representation. Instead of representing the adjacency matrix as a graph, the convolution could be applied to the adjacency matrix directly. In this case, it is essential to use another type of convolutions like, for example [Zhang and Wallace, 2015].

3.2.3 Recurrent Neural Networks

Recurrent Neural Network (RNN) dominate the field of processing sequence data in NLP for a long time. Because the already mentioned close relationship between skeleton-based action recognition and sentence classification, why not consider the RNNarchitecture also for the action recognition task?
[Ren et al., 2020] researched in a 2020 released survey multiple approaches for 3D skeleton-based action recognition. They pointed out:

> "Similarly to the standard RNN, LSTM and GRU, which introduce gates and linear memory units inside RNNs, were proposed to make up the shortages like gradient and long-term temporal modeling problems of the standard RNN. From the first aspect, spatial-temporal modeling , which can be seen as the principle in action recognition task. Due to the weakness of spatial modeling ability of RNN-based architecture, the performance of some related methods generally could not gain a competitive result" [Ren et al., 2020].

Only focusing the in this work used NTU dataset and the leaderboard on the "paperswithcode" website[3] , it can be seen that not even in the top 10, there is one single approach using Recurrent Neural Networks. While in the first years after the publication of the dataset, both, CNNs and RNNs were tested, CNNs outperform the RNNs [Li et al., 2017c, Zhang et al., 2017, Li et al.,

[3]`https://paperswithcode.com/sota/skeleton-based-action-recognition-on-ntu-rgbd` - Retrieved on: 20.09.20

2017b, Ren et al., 2020, Zhu et al., 2018]. Ren et al. admit, that because of the fact that also for RNNs, a descent modeling of the data for skeleton based action recognition is crucial for good performance, so that "RNN based methods are really weaker than CNN based methods" [Ren et al., 2020]. The difference becomes even clearer, when two approaches from the same year are being compared (see Table 3.4 and 3.5. The winner of the competition is the CNN. For this reason Recurrent Neural Networks are not considered further in this work.

2017	Cross-Subject	Cross-View
CNN(04/2017) [Li et al., 2017b]	83.2%	89.3%
RNN(08/2017) [Li et al., 2017c]	74,6 %	83.2%

Table 3.4: Comparison of CNN and RNN performing action recognition on the NTU dataset in 2018.

2018	Cross-Subject	Cross-View
CNN(2018) [Zhang et al., 2018]	88.7%	94.3%
RNN(2018) [Zhang et al., 2018]	79.8%	88.9%

Table 3.5: Comparison between CNN and RNN performing action recognition on the NTU dataset in 2018. The authors of [Zhang et al., 2018] tested their introduced techniques using CNNs and RNNs. Best results was obtained by fusing results of both architectures. A comparison of both techniques results in the CNN as a clear winner.

4 A Conceptional Approach for Skeleton-Based Action Recognition using Wide Convolutions

4.1 Adapt CNNs from Natural Language Processing

Human Action Recognition with skeleton data in sequences is a closely related task to sentence classification in Natural Language Processing. In both cases, a latent representation (embedding) of an input is created, and due to the natural behavior of language and actions, they consist of multiple smaller parts (words or poses). Especially the Recurrent Neural Network were very popular in the last years in tasks like sentence classification or sentiment analysis. Nevertheless, some approaches try to solve these tasks with CNNs [Kalchbrenner et al., 2014, Kim, 2014].

In contrast to convolutions in computer vision, in NLP, wide kernels are used for convolving over the embedding. Instead of applying a kernel only to a small region in the x and y-axis, CNNs in NLP are applied in spatial dimension overall features and in temporal dimension over a smaller window. The specific network architecture may differ from paper to paper. Still, most of these works follow the scheme described above and which is shown in Fig 4.1. Please note, that this should not be mistaken with the term *wide* from [Chen et al., 2019], where *wide* refers to the number of filters per layer.

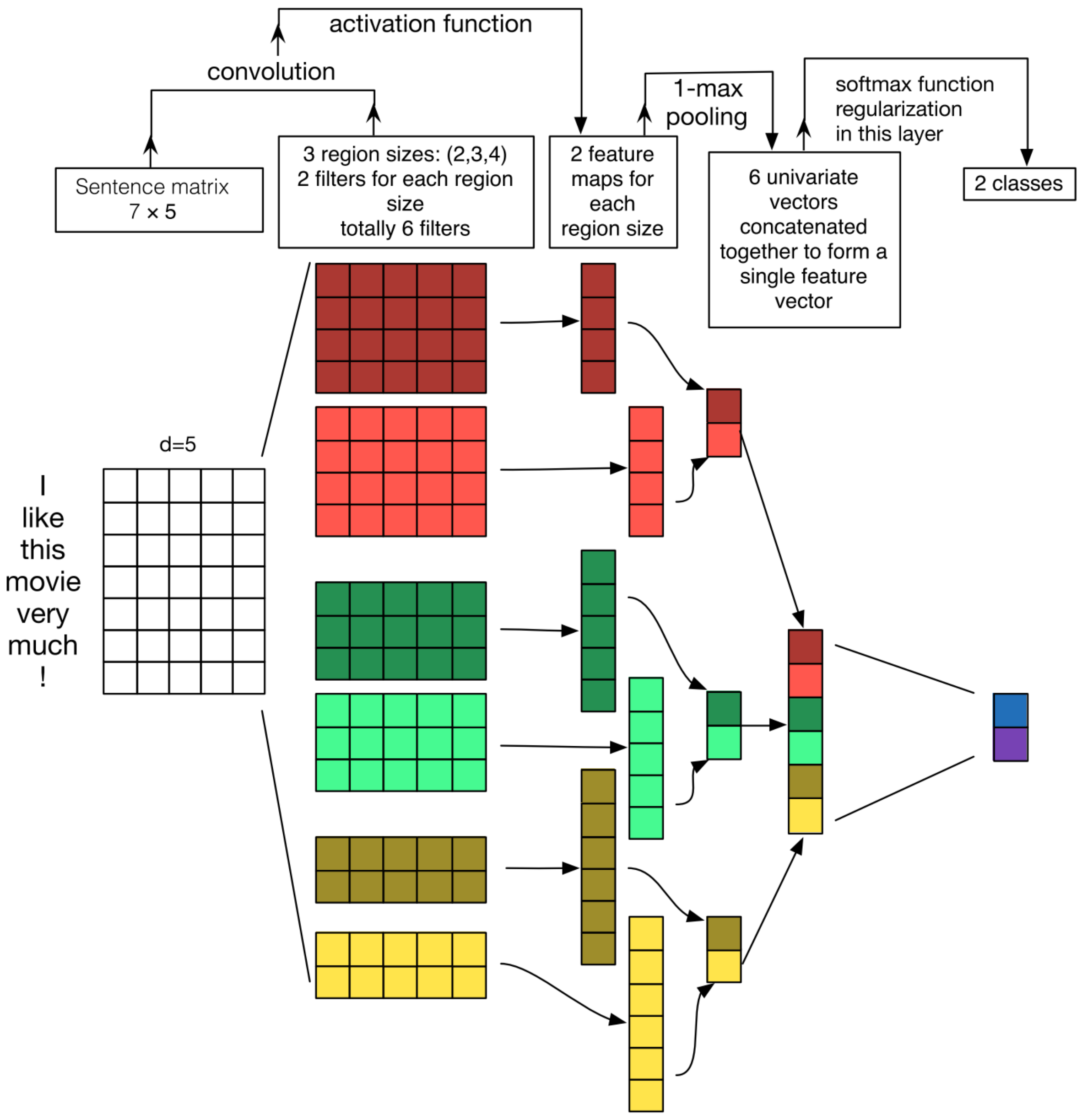

activation function
convolution
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
d=5
I
like
this
movie
very
much
!

Figure 4.1: A one-layer cnn used for sentence classification. Wide filters are used to convolve over the whole embedding as well as in the temporal dimension

[Zhang and Wallace, 2015] showed comparable results against different Support-Vector-Machines in sentence classification. But, to the best of my knowledge, no action recognition work exists that adapt this technique for skeleton-based human action recognition. Because this approach has not yet been used for such a task, no comparison can be made with the latest methods.

The main contribution of Zhang et al. was the research of which parameters does affect the network the most. They found out, that while regularization has only a small impact on the network performance, region size, the number of feature maps and the kernel size have a large effect. Zhang et al. adviced to focus on tuning them to increase performance. This is important to keep in mind, because this network will be investigated in chapter 4.4 "Network Experiments".

4.2 Data and Tools for Method Instantiation

In the following chapter, the focus lies on the investigation of techniques for action recognition under practical conditions. While the latest state-of-the-art researches are only improving techniques to reach better results on the NTU RGB+D (NTU) benchmark, these approaches are usually not transferable to a practical application. For example, the approach of [Shi et al., 2019] enables the processing of multiple skeleton representations, including 17 or 25 key points, but their approach relies on the assumption of a completely detected skeleton with all keypoints. Although today's pose estimators become better and better, this assumption does not represent the reality yet. In the following, different skeleton-based representation techniques, preprocessing, and architectures are introduced, which enable the processing of partially detected skeletons.

Due to time constraints, performance is only validated on a subset of the NTU dataset. For simplicity, six classes were picked which have inter-class overlaps in pose sequence and in appearing objects. The six selected classes are:

- **A002** Eating a Meal
- **A003** Brushing Teeth
- **A028** Phone Call
- **A029** Playing with Phone/Tablet
- **A030** Typing on Keyboard
- **A002** Taking a Selfie

Distinctions and Overlaps:

1. **A028, A029, and A032:** Inter-class variations in the pose sequences but overlap in the interaction objects
2. **A029 and A030:** Overlaps in pose movements but is different in the interaction objects
3. **A002, A003, A028:** Include sequences where a hand is moved to the head but different interaction objects are used

Picking these hard samples, it could be ensured that results reached on the smaller subset can also be reached on the "easier" big dataset.

For implementation, python as the programming language and *Tensorflow* together with *Keras* as deep learning framework is used. To prove the usability of the discussed theories in the following "Data Preparation" chapter, a self-designed CNN architecture is chosen to compare the impact of the different techniques. This network may not be the best solution for cross-subject and cross-view validation accuracy, but it is more important to have a static evaluation environment as a base where comparable results of the techniques will emerge.
The network is visualized in fig 4.2. See section 4.4.3 for specific architecture concepts.

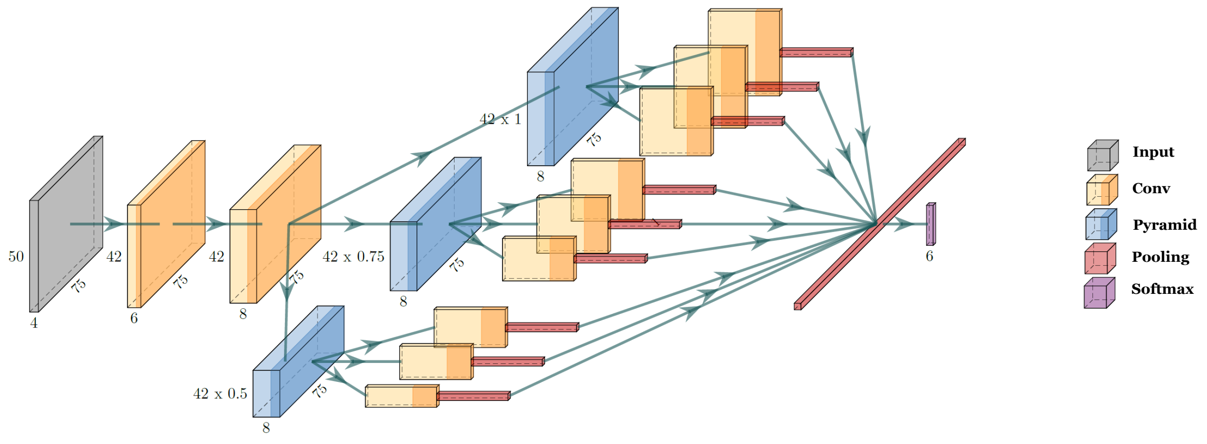


Figure 4.2: The network used for data preparation experiments first exctracts pose primitives using two consecutive convolutions followed by wide convolutions

4.3 Data Preparation

By using skeleton-based action recognition algorithms, high variances appearing on images usualy are avoided. Variances in images result for example from the camera sensor, the lighting situation or the scene as such.
However, variance also exists in skeleton data, but in a reduced form. The data can differ in their viewing angles, the viewpoint, the speed of the performed action as well as the subject that executes the action. A few of these variances can be handled by good data preparation. In the following, different normalization approaches are introduced and their influence is discussed.

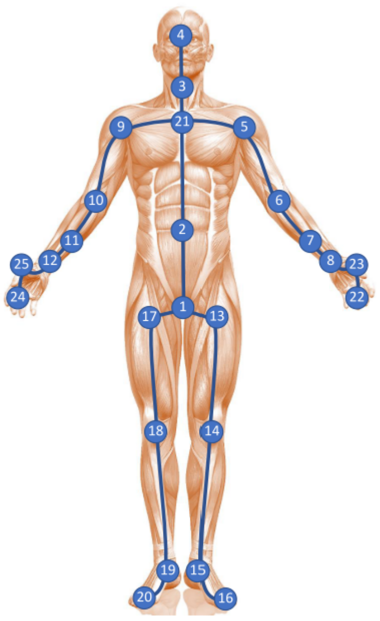


Figure 4.3: 25 skeleton key points extracted by Microsoft Kinect V2 [Shi et al., 2019]

4.3.1 Viewpoint Normalization

In most scenarios of the Inferics GmbH, the camera is fixed over time, and the subjects performing the actions are free to move around in the room. The most common way to deal with viewpoint variance is to transform the pose from a world coordinate system to a body-related coordinate system relative to an anchor point of the human itself. [Shi et al., 2019, Yan et al., 2018] both use the spine as the origin of the new viewpoint invariant coordinate system. They then calculate the difference vectors between each keypoint and anchor point and take the result as a new joint coordinate.

[Kim et al., 2019] suggest the addition of essential joints as reference vectors, like, for example hands. This would improve the recognition performance. But it also means that in addition to the 25 difference vectors between joints and spine, 50 more vectors need to be calculated (25 for each hand). The parameters a network needs to learn are increasing drastically. The complexity of networks is referred in the following as FLOPs.

Both representations could lead to problems, if single important joints are missing. This is not uncommon for today's pose estimators because single joints are not recognized correctly due to occlusions. To deal with such issues, it could help to take each joint as a reference vector and calculate the difference vectors from each joint to each joint. This would be the way to go, when applying such techniques in real-world applications, and enough time for training is available. For a more data-saving representation, it is sufficient to consider only the vectors from joint **a** to joint **b**, but no further the inverse vector from **b** to **a**. Thus one obtains a total of 300 difference vectors. Nevertheless, with this representation, the worst-case scenario may occur with the network only learning one or two discriminant joints. If these are omitted, the action cannot be recognized correctly. To avoid such a behavior, regularization at the wide kernels could be used.

For the following experiments, the spine and both hands are chosen as reference points. It enables faster network training and better evaluation of the convolutional feature maps than 300 difference vectors.

4.3.2 Orientation Normalization

Another degree of freedom in skeleton-based action recognition is the orientation of the human skeleton. The latest works simply rotate the vector connecting the left and the right shoulder and the spine of each frame parallel to the x and respective the y-axis. This can be imagined as a person hanging on a wall by the shoulders. The movements of the limbs then look like a pedaling. On the one hand, this enables a better representation of actions performed only with hands, while sitting or standing still, but on the other hand makes it harder to recognize actions where a movement in space is essential, like dancing or falling to the ground.

This common approach is not used in this thesis. Instead, a more natural representation would be to translate the shoulders only around the z-axis perpendicular to the ground. This can be done by taking combinations of key point pairs. For example, left and right shoulder, left and right hip, left and right eyes, or left and right ears. One of these pairs is used together with two points lying in the x, y-plane, and the Kabsch algorithm [Kabsch, 1976] to rotate the body around the z-axis. Instead of "moving" the person with 3 degrees of freedom (x,y,z), the person is only rotated around the specific axis. This does not change the formations of the pose itself.

If movements in space are essential for a specific action, rotating all frames separately could lead to the loss of important information. The loss of information is much more significant, when both, the shoulders and the spine, are rotated into the corresponding axes instead of the new own developed method.

4.3.3 Subject Normalization

To deal with persons of different size, action recognition can be improved by normalizing the limbs of a person, or respective the difference vectors. SOTA algorithms scale each channel (x,y,z) to form a uniform distribution. Unfortunately, they do not mention the interval of the distribution in their work. A scale into the range [0,1] would not make any sense, because this might set important, but very small, values to 0. If values in the resulting representation are equal to 0, they are not considered by a convolution operation. (See also chapter "zero-padding" for further explanations). A scale between a small number > 0 and 1 could be one solution. However, by using difference vectors instead of absolute world coordinates, the normalization of the input values has little or no effect on the results of the classification as training results showed. Another positive side effect of normalization is, that the weight and bias values are kept small. In a well designed deep neural net, these values remain in a similar range over the whole net. This in turn, helps to avoid exploding gradients. By using difference vectors, the net already has values in a small range so no additional scaling is required to achieve this effect.

4.3.4 Sequence Lengths

Sequences of variable length are characteristic for human actions. An investigation of the action sequences showed that the length not only depends on the performed action, but also on the person performing the action. Person A could perform an action "Eating" faster than person B. Because of this, we need to deal with different sequence lengths for the same action performed by multiple subjects. Fig 4.4 shows the distribution of each of the six used action classes:

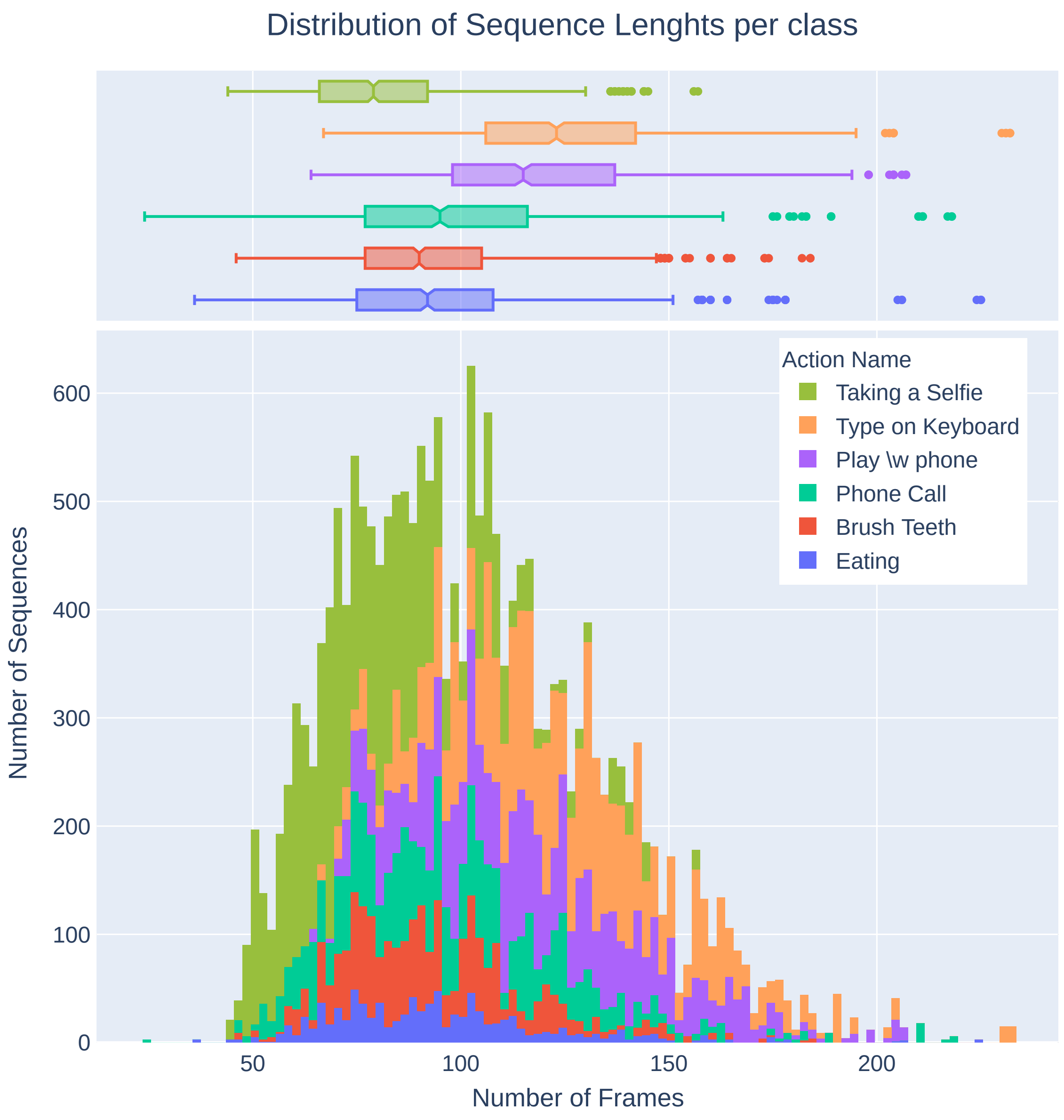


Figure 4.4: NTU RGB+D Sequence Length Distribution

The sum of all actions has a mean of around 100 frames per sequence. However, if you look at the lengths of "Taking a Selfie" and "Type on Keyboard", we see that the lengths are stretched in two different directions. To train the later network with mini-batches, two options are available: Using Tensorflow's *Ragged Tensors* or padding the sequence (see fig 4.5). *Ragged Tensors* can keep multiple sequences with variable lengths. Unfortunately, the used Keras Layer API does not support these *Ragged Tensors* natively. To be able to train the net with unmodified sequence lengths, only the option of using single sequences instead of a batch remains. But this leads to a problem with all gradient-based optimization algorithms of deep neural networks. Because they need a mini-batches that better represent the diversity of the whole data set than a single sample to compute accurate gradients.

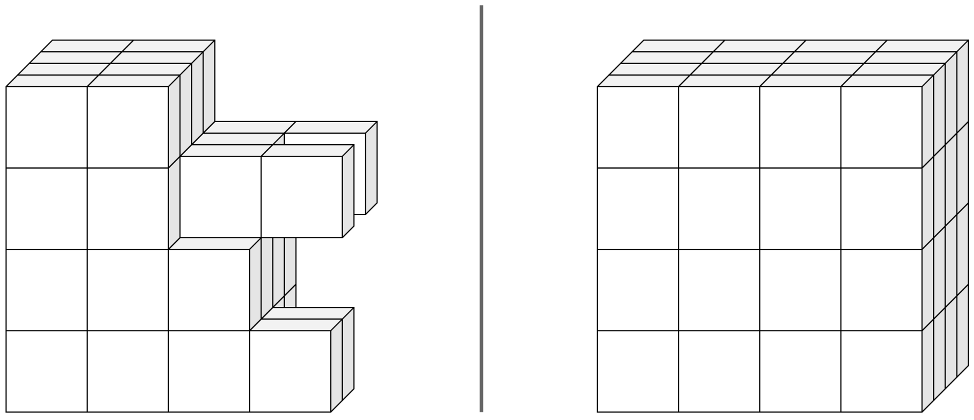


Figure 4.5: Ragged Tensor (left) vs. Regular Tensor (right)

The other option, regular tensor reached with padding, manipulates the sequence length so all the sequences can be concatenated to a batch with the same amount of values for each sample. Because of the issue of using single samples in combination with the behavior of gradient-based optimizers, *Ragged Tensors* with a batch size of 1 are no option here. Instead, sequence-padding is a well-known workaround already used for multiple years in other sequence data domains like NLP. Not only for the training process but also in a productive environment, typically, a frame buffer of the last frames is kept and processed by an algorithm. So padding would not only improve the training process but also match the typical procedure of sequences in productively used products.

To pad action sequences, there are multiple options:

(A) Padding a sequence with a fixed value

(B) Padding a sequence with the sequence itself

(C) Padding a sequence with the last frame

(D) Resize the sequence or padding with zeros

Optimal Sequence Length

A closer look at each action gives more insights about intra-class and inter-class variance. Table 4.1 contains the key indicators of each action class. The Min and Max values differ too much from each other to use them as a good indicator. By calculating the 10th and 90th percentile, outliers are ignored, and a pattern can be seen. Both numbers are different by a factor of 2. Instead of a range from 36 to 232, we can limit the range from 57 to 174. To get a round number, 170 frames as maximum sequence length were chosen. All sequences with more frames are trimmed, and all sequences with less than 170 frames are padded regarding a specific strategy.

Name	Min Frames	Max Frames	Mean	10 Percentlile	90 Percentlile
Eating a Meal	36	225	94.85	66	144
Brushing teeth	46	184	93.56	69	133
Phone Call	24	218	98.73	65	147
Play with Phone	64	207	118.67	83	168
Typing on Keyboard	67	232	125.43	92	174
Taking a Selfie	44	157	80.73	57	116

Table 4.1: Seuence length distribution of the six chosen classes

This assumption is investigated in the following, regarding padding strategies **A,B,C** and **D**. For evaluating these padding techniques, it is important to keep in mind that most layers in the later introduced networks will be convolution layers, and as input, a batch of 3-dimensional tensors will be processed. The first dimension is the time t, and the second dimension contains the features which could be the limbs itself or another skeleton representation. The last dimension contains the x, y, z coordinates, plus a confidence value c. This can be seen as an analogy to computer vision representation of an RGB image with a fourth alpha channel.

To match a chosen sequence length, frames containing less values are padded according the chosen strategy to let the sequence grow in the temporal dimension. [Dwarampudi and Reddy, 2019] researched the impact of padding on CNNs and LSTMs. They found out that LSTMs are very sensitive with respect to the side where values are padded. While their simple baseline LSTM reach 80% on train and test dataset with padding in front (pre padding), the accuracy collapses to 49% train accuracy and 50% validation accuracy with post padding, which is very bad for a 2-class problem. In their advanced studies, they showed that the side of padding does not necessarily have any impact on CNNs. The CNN reached 74% train, as well as validation accuracy for both pre and post padding and, shows higher robustness. It is essential to know this aspect, if the classification algorithm would be changed in the future.

A: Zero-Padding

In a zero-padding strategy, tensors are padded to the desired length by filling the tensor with zeros. Theoretically, zero-padding does not influence the resulting feature map of a convolution. If we calculate the dot product between the padded input I and the convolution kernel K, the result does also have only zeros. This process is visualized in fig 4.6. But this consideration is only partially correct. In the used Keras implementation, the Conv2D kernels also have a bias value that is added to the feature map. So the output of such a layer is just the bias value. This behavior can be fixed by removing the bias from the convolution layers. Even when the bias is removed, we could get negative effects at the edge between the end of a sequence and the beginning of the padding. An investigation showed (fig 4.7) that the bias removal leads to a lower validation accuracy. This means that the bias is more important for the original parts of the sequence than for the padded parts.

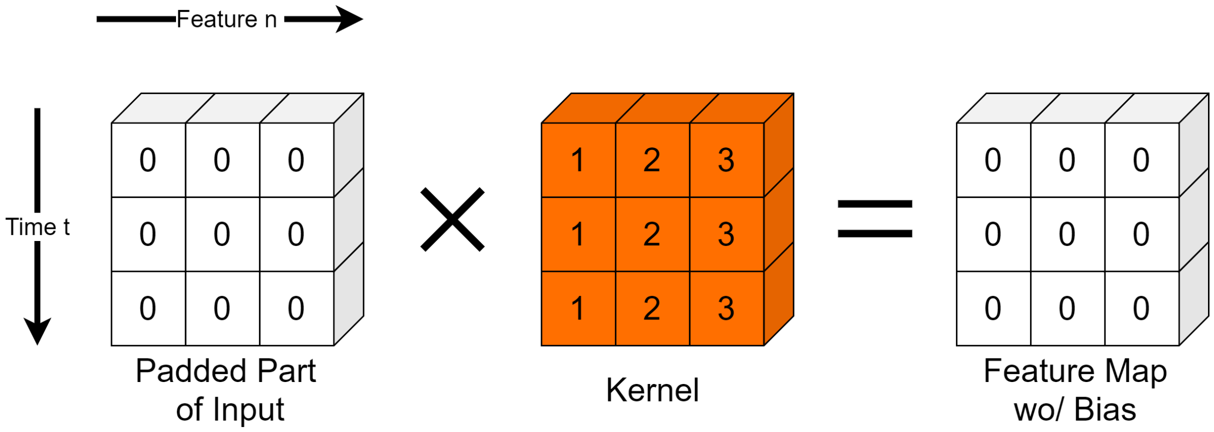


Figure 4.6: Impact of zeros at convolution

Reducing the sequence length using a resolution pyramid, the zeros and the real values are included in the calculation of the new values, which could significantly distort the sequence. So if the original sequence is compressed too strongly in the temporal dimension, the zeros are a significant disturbance factor at the border between original and padding. However, this error would also occurs when padding with the sequence itself and also could have an unforeseeable effect.

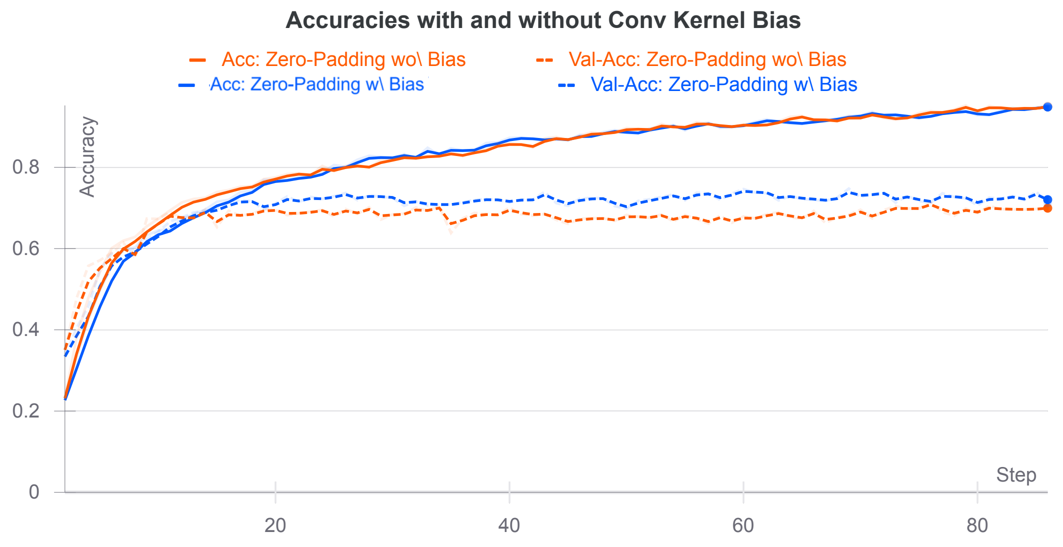


Figure 4.7: Zero-Padding: With bias and without bias in convolution layer

B: Padding with the sequence itself

Padding with the sequence itself is a common strategy in skeleton-based action recognition [Shi et al., 2019, Yan et al., 2018] . It is done by repeating the action sequence again and again until a specific number of frames is reached. In the following, the strategy is referred as "sequence-padding". In contrast to zero-padding, this strategy would heavily affect the convolution results. Imagine a perfect kernel that describes a certain action sequence sufficiently. If this kernel is moved over the action sequence, which indicates the similarity of the learned action sequence to the sequence cutout that was just convolved with the kernel. In a zero-padding scenario, it can be expected to have strong activations in the first part of the sequence and nearly zero activations in the last parts, while sequence-padding will have high activations at the beginning as well as in the end. Indeed, this affects the classification results. In fig 4.8, the zero-padding strategy is compared with the sequence-padding strategy. The network trained with the sequence-padding strategy reaches a slightly better accuracy score.

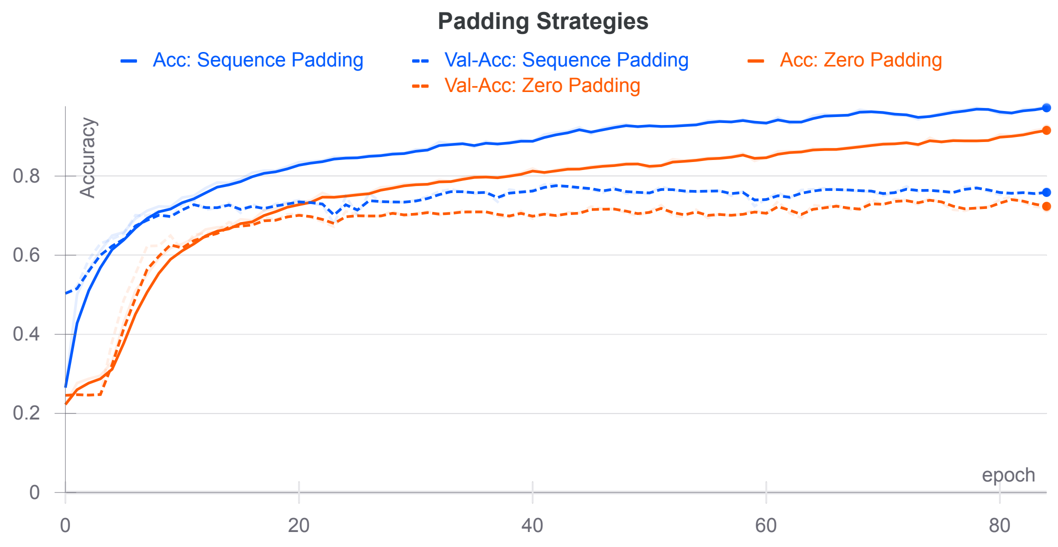


Figure 4.8: Zero vs. Sequence-Padding

Although sequence-padding leads to better classification results on the train and validation set, this padding strategy makes no sense for a real-world scenario. In computer vision applications, image buffers that contain the last *n* frames are often used. They hold a continuous sequence at every time step. Also the padding with the sequence itself leads to a hard cut at the end of the sequence, which is not representative for a convenient use-case.

C: Padding with the first or the last frame

This strategy simulates a still image at the end of the sequence. In contrast to padding with a sequence, appending the last frame of a video would not result in higher activations at the padded parts. The advantage to zero and sequence-padding is that hard cuts at the end of the sequence are avoided. In theory, this padding is closer to the real-world scenario as the previously introduced mechanisms. But, if the sample has only a few frames, the padding will lead to a heavily weighted rigid pose. Imagine the input sequence "Eating a Meal" has a length of 25 frames containing the movement of the right arm from the plate to the mouth. Then, the last frame, which contains a very discriminative pose ("fork at the mouth") is padded about 100 times. We need to assume that the network only learns this rigid pose that would lead to a bad generalization. While in other cases, like, for example, "phone call", padding of the last frame, "hold the phone to the ear" seems to be natural. However, we need to assume that this technique leads to a classification of just the noise, here referred to as a very discriminative last skeleton pose instead of the individual sequence.

D: Resize the sequence or padding with zeros

In computer vision, images are often resized before processed by neural nets. A common strategy is "resize with padding". If an image is bigger than the expected input, it will be downsampled. If the image is smaller than expected, the image is resized with padding at the edges to avoid upsampling artifacts.

This is a combination of the previously introduced zero-padding, which does not affect the classification results negatively, and a rescale of the sequence without losing information. The last statement is only correct if the Nyquist-Shannon sampling theorem is respected. The Nyquist-Shannon theorem states, that function $x(t)$ with a highest frequency f can be perfectly reconstructed by sampling with a minimum frequency of $1/2f$. Considering one of the fastest actions as example: Clapping. Clapping is performed with a frequency of round about 4 claps a second (4 Hz; 250 ms). If a video is captured with 30 fps, the duration of one frame is 33 ms. Sampling all sequences to have a length of 60 frames compress the longest sequences by a factor of one quarter. So the Nyquist-Shannon theorem is fulfilled even for the outlier sequences with a length of 232 frames (see table 4.1).

In figure 4.10 four downsampling techniques provided by Keras are visualized. The inputs of the downsampling function are two sequences (orange and blue) containing 120 2-dimensional joint coordinates, x, and y. While the methods "Bilinear", "Nearest Neighbour" and "Lanzcos3" calculate good approximations, the "Gaussian" sampling does not match the original curve. This is an issue of the Keras implementation where the Gaussian kernel uses neighbors in x as well as in y-direction, even if only one dimension should be resized.

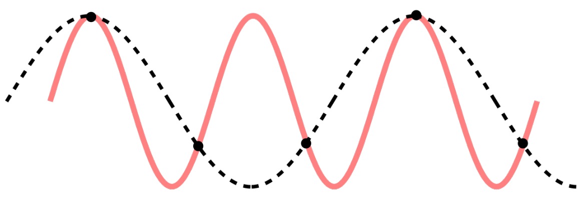


Figure 4.9: Nyquist-Shannon sampling theorem: Bad sampling rate creates wrong sequence

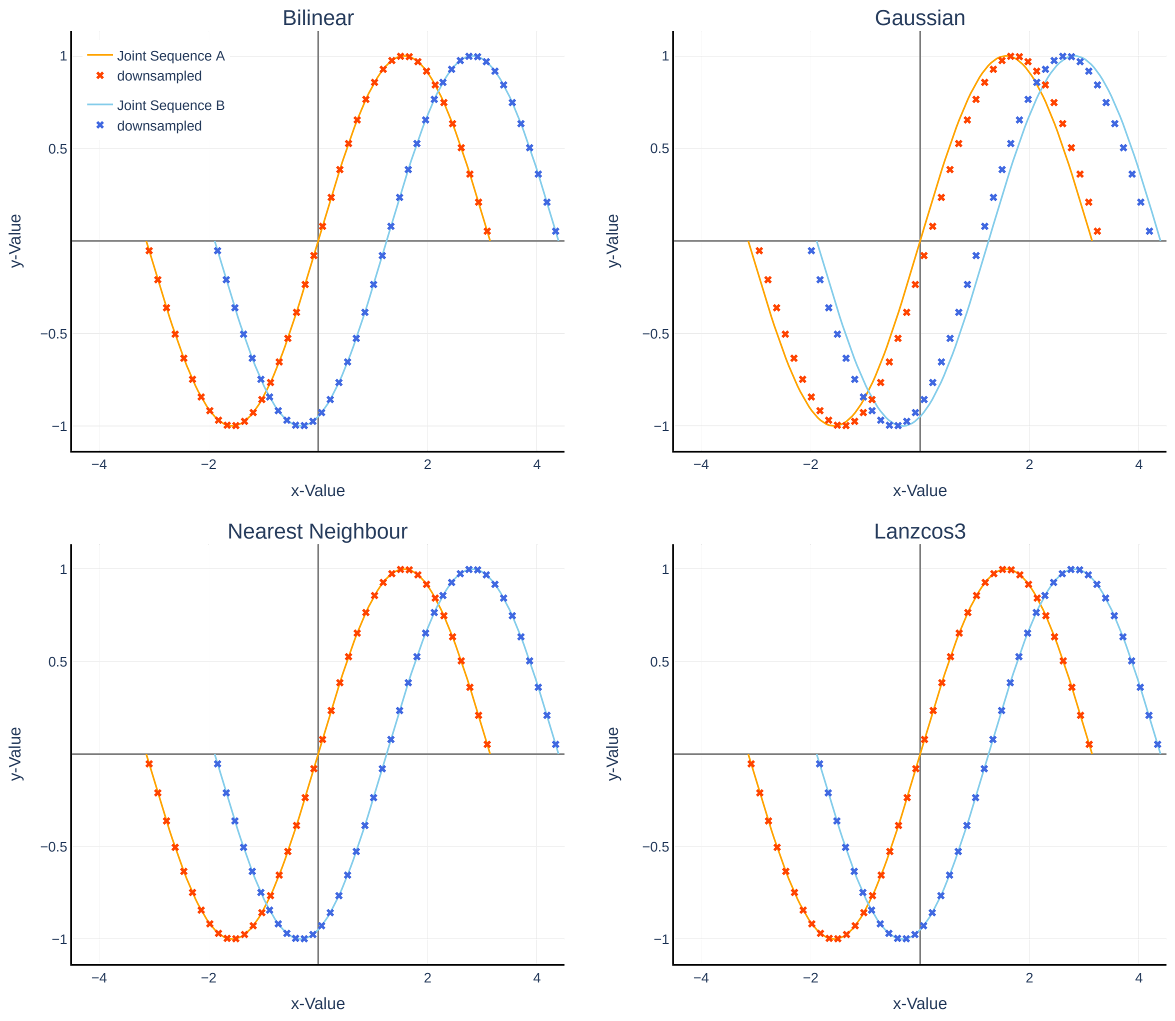


Figure 4.10: Four different sampling strategies provided by the keras framework were investigated.

Investigations showed a performance increase while decreasing the parameters of the network, which result in faster inference. For a fair comparison, a few training parameters need to be specified:

- The used networks have on top of the convolutional feature extractor the earlier introduced wide convolution with a temporal receptive field of 15 frames. The layer holds 90 different kernel filters.
- With a hyperparameter tuning, 170 was determined as the optimum value for the methods A, B, and C with regard to the maximum sequence length. For the "Resize with Padding" strategy, 53 was determined as the best sequence length. This difference leads to fewer network operations.

Further reasearch indicated that with a good hyperparameter tuning, more than 80% validation accuracy on cross-subject set could be reached. But for a fair comparison, figure 4.11 visualized randomly picked runs. While the accuracies of the four methods are very similar, it is remarkable that the "Resize with Padding" methods need only one-third of the parameters.

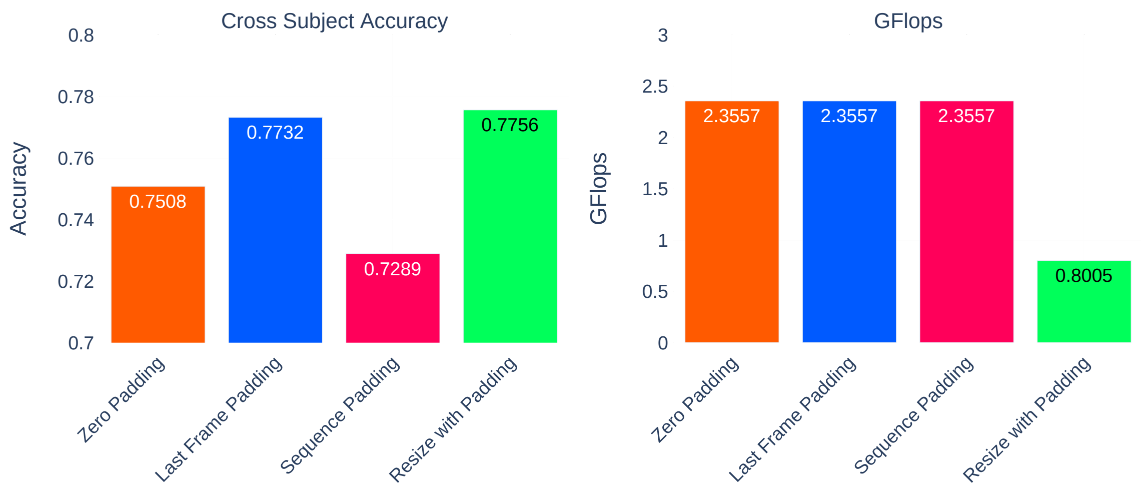


Figure 4.11: Padding Strategies: Accuracy vs. FLOP

4.3.5 Feature Extraction

Chapter 2.3 introduced the desired inference pipeline, which includes a pose estimator and an object detector as feature extractors, followed by data preprocessing and inference of the action classifier network.

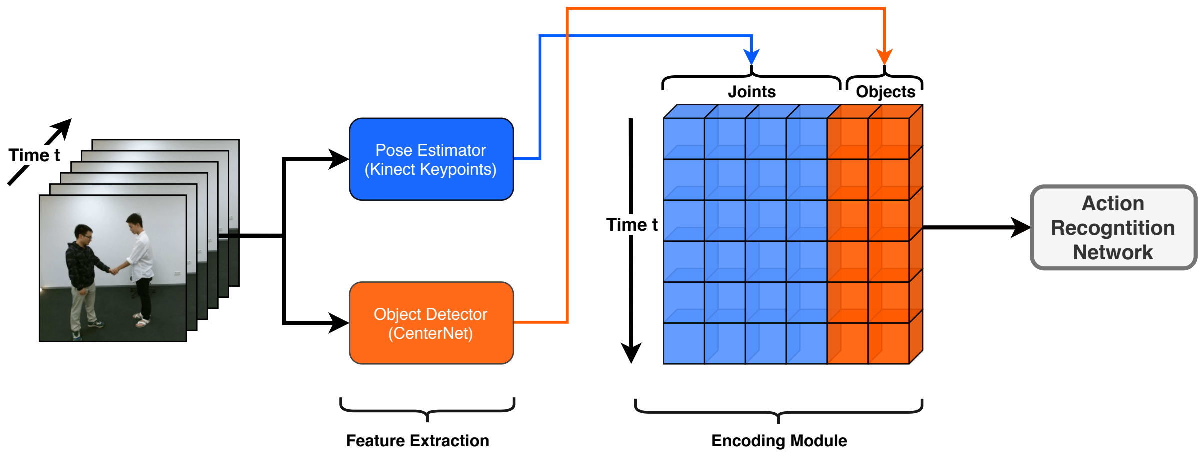

Time t
Pose Estimator (Kinect Keypoints)
Object Detector (CenterNet)
Joints
Objects
Time t
Action Recognition Network
Feature Extraction
Encoding Module

Figure 4.12: Pipeline structure as introduced in chapter 2.3.

Figure 4.12 visualizes this process again. The focus in this work lies in the action classification itself. For skeleton data, the skeletons provided by the NTU dataset are used instead of the in figure 4.12 visualized pose estimator, but this part could be easily exchanged later. To evaluate if context objects influence the action recognition task positively, objects need to be detected first. Therefore, a variant of the so-called *CenterNet* [Duan et al., 2019] is used to detect objects in the 2D RGB sequences of the dataset and transform them into the 3D world coordinate system with the camera matrices. The *CenterNet* reaches superior performance on the COCO object detection task with 47% *Average Precision* and 37.5% *Average Recall* and outperforms the well-known *Mask R-CNN* [He et al., 2017]. In contrast to other approaches, it does not predict the bounding boxes as a pair of keypoints. Instead, a low-level keypoint detector is used to find the mid of an object, and this key point forms a triplet together with both corner points of the bounding box. Duan et al. showed an improvement in both precision and recall.

TensorFlow provides in their object detection API model zoo a few pretrained object detection models, which can be used for easy and fast deployable inference. In this work, the "CenterNet HourGlass104 512x512" – model is used to extract context knowledge in form of objects. Keypoints detected by the pose estimator and the object detector are then fused. The fusion done by concatenating the detectios. The final input tensor for the action recognition network hold x,y,z coordinates and a confidence score output by the network over time (see figure 4.13. In the most scenes, multiple instances of the same object are detected. This is a problem because the neural nets usually expects inputs of fixed sizes but how can the input size being fixed?

In theory, if a user interacts with an instance of an object, the interaction instance is closer to the human than the other instances of the same object class. A good approach would be to take the closest object instance to the user. For this purpose, the distance between each detected object and the commonly involved limbs like hands, feet, and head is calculated. The objects with the smallest averaged distance is chosen as interaction object.

The COCO dataset provides for the earlier chosen action classes "Eating a Meal", "Brushing Teeth", "Type on Keyboard", "Taking a Selfie", "Phone Call" and "Play with Phone" the following matching object classes:

Fork	Knife	Spoon	Bowl	Banana	Apple
Sandwich	Hot Dog	Pizza	Donut	Cake	TV/Monitor
Laptop	Mouse	Keyboard	Cell Phone	Toothbrush	

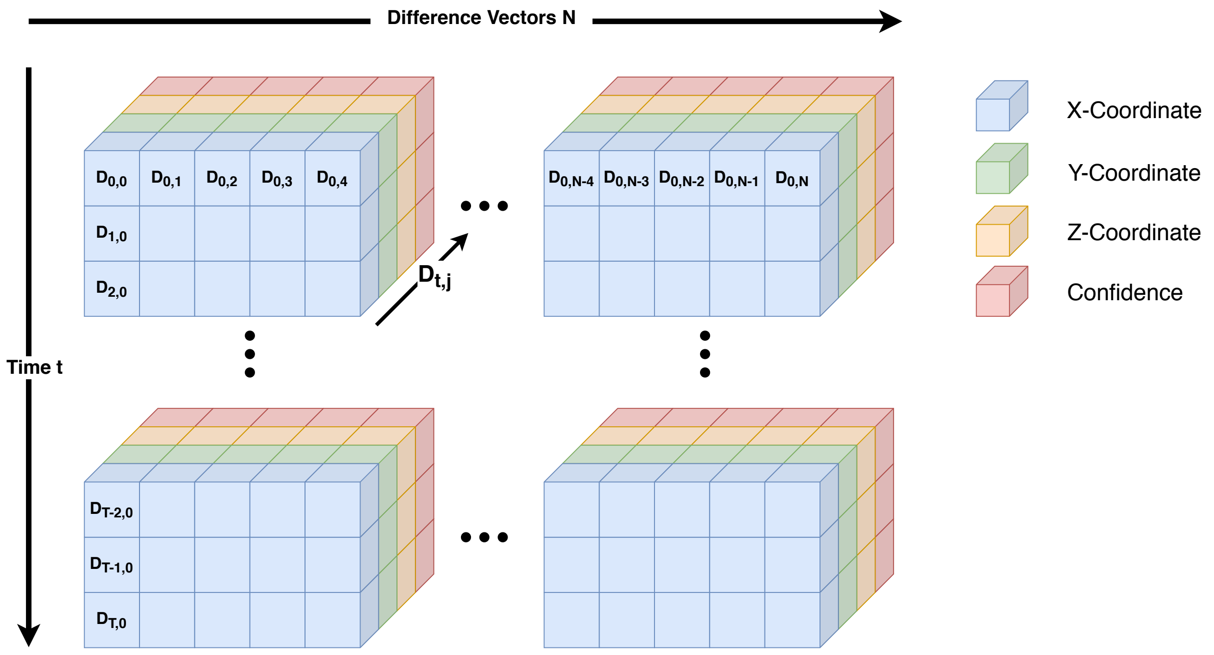


Figure 4.13: The 3D input tensor holds a pose representation at each time step.

The final input tensor holds a difference vector at every position. This ensure a normalized skeleton representation (See chapter 4.3.1. The amount of difference vectors depends on the chosen strategy introduced in 4.3.1 "View Point Normalization" and on the amount of additional key points used. For example, figure 4.13 shows the result of the method using the mid spine with $D_{0,0} = J_0 - J_{Spine}$ and $J_0 = (x_0, y_0, z_0)$. The confidence value is added regarding a specific aggregation strategy. Currently implemented are the minimum, the maximum, and the mean strategy. Minimum and maximum strategies take the smaller or larger of the keypoint confidence values. Mean calculates the mean of both. Each row can be seen as a frame from time step t. The entirety of all frames forms a sequence, which contains the information of a pose over time. The depth dimension holds the x, y, z, and the confidence value c of the vector. If just the 25 skeleton joints are used, the input has a shape of $(T, 25, 4)$. Adding the mentioned object classes, 17 additional difference vectors are calculated.

Following the approach introduced in [Kim et al., 2019], left and right hand are used additionally as reference points. [Kim et al., 2019] have shown that the enrichment of the human pose by hand keypoints leads to an overall improvement when objects are involved. The disadvantage of taking the hands as additional reference points, is that the number of difference vectors is tripled.

4.3.5.1 Issues

To be able to use objects as context knowledge, they have to be detected in 2D by an object detector as described above and then transformed into 3-dimensional space using the camera matrix, that transforms the RGB into the depth map. It provides the corresponding x, y and z coordinates of the world coordinate system. Unfortunately, this could not be implemented as planned for several reasons:

- [Shahroudy et al., 2016] provides RGB video sequences and corresponding depth maps from a Kinect V2 camera. But they did not save the camera matrices, which should enable the transformation from one to another camera coordinate system, which is necessary to calculate the world coordinates based on the detected key points of the object detector.
- In Theory, it is possible to calculate an approximation of the transformation matrix by the stored skeleton keypoints. These approximations enables the calculated of a projection from 3D points into the RGB frames. This need to be done for each corresponding pixel and each camera setup which is very time consuming. Therefore this part is not covered in the thesis

A workaround to investigate the impact of objects to skeleton-based action recognition could be using the provided skeleton keypoints in 2D RGB iamge as well as the detected objects and show a positive impact using only two-dimensional data.

4.4 Network Experiments

To the best of my knowledge, no prior skeleton-based action recognition experiments using the so-called "wide convolutions" exist. The goal of this chapter is not to reach the top of the NTU RGB+D leaderboard and compete against the other SOTA approaches. The goal is to experiment and discuss different design decisions for the deep neural net architecture. Further, the most important goals are to show that in general, the wide convolutions from sentence classification are adaptable for skeleton-based human action recognition. At the same time, the number of arithmetic operations (Flops) should be kept as low as possible. Due to time constraints, this work does not claim to be complete for all aspects investigated. Further ideas, which unfortunately cannot be investigated due to this time limit, are listed in the chapter 6 "Outlook".

The models which are tested and introduced in the following using data created by combining the best approaches introduced in the previous chapter. The experiments are run with the following data preparation techniques:

- Difference vectors between all key points and the reference points spine, left hand and right hand.
- The data is rotated into the x,y-plane using the Kabsch algorithm [Kabsch, 1976].
- "Resize with padding" is chosen to save the number of parameters by keeping high accuracy. chosen resize method is bilinear interpolation. All sequences with less than five frames are rejected due to the high probability of being broken sequences. 50 was chosen as the max sequence length.
- Data is not being scaled

Other training routine parameters are fixed over all settings to keep the results comparable:

- Batch Size: 128
- Optimizer: Adam with Amsgrad [Reddi et al., 2019]
- Loss: Focal Loss [Lin et al., 2017] / Categorical Crossentropy
- Learning Rate: $1 * 10^{-3}$
- Training Epochs: 85
- Action Classes: 6 (See introduction of chapter 4)

The starting point of the architecture experiments is the baseline model with wide convolutions discussed in 4.1. The Baseline model with filter heights of 2, 3 and 4 with 2 filters each kernel reaches a CS score of 45.66% and a CV score of 55.64%.
Even after compressing the sequence to only 50 frames, this length still differs too much from the original use case sentiment analysis, where input sequences have a mean length of $\tilde{1}6$ words [Zhang and Wallace, 2015]. So the receptive fields of convolution filters with a height of 2, 3 and 4 cannot capture the whole context of the sequence. The max-pooling over the whole feature map, which break down multiple values into a single one, completely destroy the temporal information. Doubling the kernel heights and triple the number of kernels, the CS score can be increased to 61.97% and the CV to 66.28%. All experiments using the baseline are summarized in Table 4.2.

Kernel Heights	Filters per Kernel Height	Cross-Subject	Cross-View	Flops
[2,3,4]	2	44,66%	55,64%	10.7K
[4,6,12]	6	61.97%	66.28%	79.4K
[4,6,12]	12	**71,13%**	**73.73%**	158.8K
[8,12,24]	6	16.7%	51.16	158.9K
[8,12,24]	12	70.95%	66.98%	318,0K
[6]	24	65.43%	72.2%	86.7K
[9]	24	58.82%	66.09%	129.8K
[12]	24	64,56%	50.79%	173.0K

Table 4.2: Results of the Baseline Model

Increasing the capacity also increases the accuracy, but with the trade-off of multiplying the weights. If the chosen kernel height is too big, it seems that the network is unable to learn accurate representations. All networks reach a slightly different training accuracies of ±5%. This could be an indicator, that the capacity of the network is fully utilized. In order to further improve the results, the architecture must be rethought. Where are its weaknesses, and what should be retained?
In a short ablation study, it was examined whether similar accuracy values could be achieved with the kernel of a single size. However, it turned out that the addition of further kernel sizes has a positive effect on the classification result.

4.4.1 Resolution Pyramids

Already discussed in the data preparation chapter, a high intra-, as well as inter-class variance in sequence length exists. A similar variance appears in computer vision object detection algorithms. Objects appear in small or big size depending on their distance to the camera. One technique that should help to deal with these is the concept of resolution pyramids. While images are scaled down in x and y direction, in action recognition, just the sequence length (y) should be modified. A sweet spot for the maximum down sampling factor would be using the factor of two. This value is the factor between the 10th and 90th percentile of the sequence lengths (see section 4.3.4. The implementation of the resolution pyramid principle in this work enables an individual definition of the number of pyramids. Each is created by a factor of maximum two. So if three is chosen as a resolution pyramid number, the algorithm outputs the original sequence (factor 1), the sequence scaled down by a factor of 0.75 and the pyramid that has half the size of the original (factor 0.5). Figure 4.14 visualizes such a sampling process.

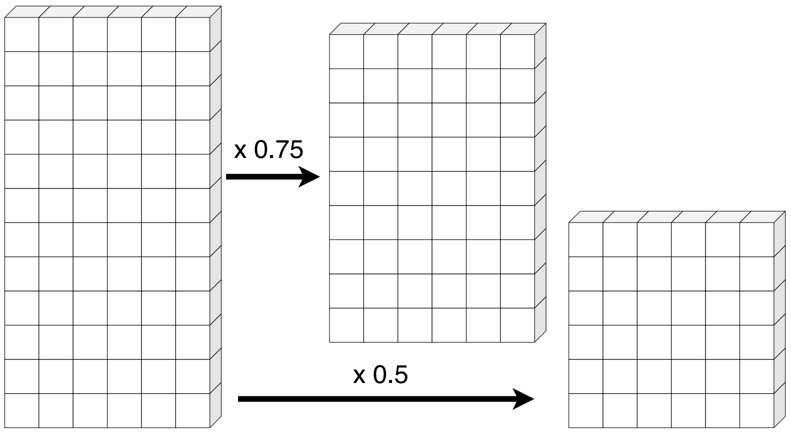


Figure 4.14: Resolution Pyramid with three different scales

The architecture including the resolution pyramids is visualized in figure 4.2. Additional to the baseline, a part is added that creates the resolution pyramids based on the given parameters. Via these parameters, the sampling method, the number of resolutions and whether to use an antialias filter when downsampling the sequence can be adjusted.

Experiments have shown, that on the one hand, increasing the number of resolution pyramids also increases the number of Flops drastically, but it leads to an improvement in terms of CS accuracy and CV accuracy. Nevertheless, a further increase of the amount of pyramids might not necessarily lead to an improvement. Probably the network cannot fully utilize them because too many parameters needs to be learned.

4.4.2 Sharing Weights

In theory, a specific action **A** consists of the same primitives, independent on the performing subject or the viewpoint. Imagine an action that looks like a sinus wave (see figure 4.10). If different subjects perform this action, the sinus-like wave varies in amplitude and frequency. Instead of normalizing the sequence in length, the sequence should be provided with different scales to the convolutional layers. The image pyramids provide these different scales. Instead of learning the convolution kernels separated for each resolution pyramid, the weights are shared between each pyramid. By saving the weights, additional filters can be added to further improve the classification performance. Table 4.3 compares methods with non-shared weights and shared weights. By sharing the weights between the convolutional layers but keeping the same kernel sizes and the same number of kernels, it is not possible to reach similar good results. But increasing the amount of filters of the weight sharing version, similar good results are reached on Cross-View and even outperform the old variant on the Cross-Subject validation set.

Thus, similar good results can be reached using only 66% of the weights. Additionally, the number of resolution pyramids could be increased by a multiple without a noticeably increase in the number of network parameters to be learned.

Kernel Heights / N Filters	Share Weights	No. Resolution Pyramids	Cross-Subject	Cross-View	Flops
[4,6,12]/12	No	1	71,13%	73.73%	158.8K
[4,6,12]/12	No	2	75.02%	80.54%	318.1K
[4,6,12]/12	No	3	75.74%	**81.8%**	476.9K
[4,6,12]/12	No	4	75.95%	80.91%	635.9K
[4,6,12]/12	No	5	**76.47%**	81.65%	794.8K
[4,6,12]//12	Yes	3	71.8%	78.69%	159.9K
[4,6,12]//24	Yes	3	**76.83%**	**81.59%**	319.4K
[4,6,12]//24	Yes	5	75.38%	79.32%	321.1K

Table 4.3: CS and CV Accuracy using shared weights vs. seperated weights

4.4.3 Keeping Temporal Information and Extracting Human Pose Primitives

In the previous chapters it was shown, that the concept of wide convolutions is applicable for skeleton-based human action recognition. Therefore, this concept is now improved.

The first improvement being investigated is to exchange the max-pooling layer, that pools the whole feature map from the wide convolutions into a single value. Using the baseline max-pooling over the whole sequence results in a complete information loss at the part of the sequence, where the learned features are detected. Two probably better approaches are investigated: Using learned convolutions to decrease vector size and decreasing the max-pooling size.

If the result of one single wide convolution would be broken down, a matrix holding the joint information for each time step is transformed into a single vector. This vector contains the activations at each convolved time step. Reducing this vector two a single value might be insufficient. Different pooling strategies are investigated (see figure 4.15).

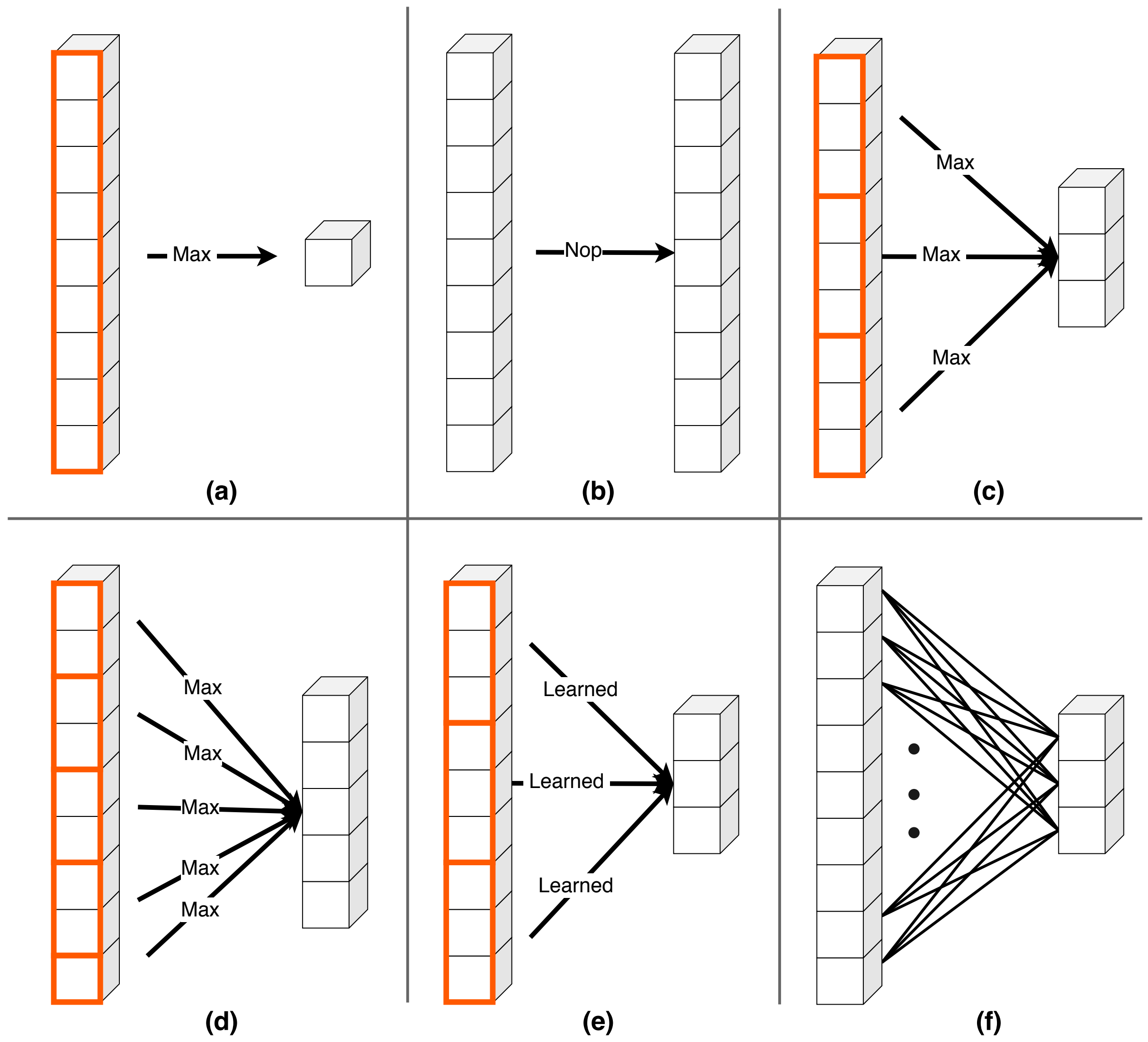


Figure 4.15: Different pooling strategies for less information loss. From left to right, from top to bottom: **(a)** Original, **(b)** No Pooling, **(c)** Lower Pool Size (3), **(d)** Lower Pool Size (2), **(d)** Conv Pool (3), **(e)** Dense Layer

Table 4.4 shows the results of the different strategies. Apart from the increase in FLOPs, no noticeable improvement is observed. The minimal differences are probably caused by stochastic circumstances during training. In fact, it would be expected, that the variants "Conv Pool", and "Dense Layer" should be an improvement over the original pooling because they have more trainable parameters which is not the case.

Kernel Heights / N Filters	Pooling Strategy	Cross-Subject	Cross-View	Flops
[4,6,12]/12	Original	76.83%	**81.59%**	319.4K
[4,6,12]/12	Lower Pool Size (3)	**77.04%**	80.22%	342.2K
[4,6,12]/12	Lower Pool Size (2)	74.1%	77.85%	356.5K
[4,6,12]/12	Conv Pool	75.2%	78.74%	352.8K//5.83
[4,6,12]/12	No Pooling	75.74%	73.21%	397.2K
[4,6,12]/12	Dense Layer	75.99%	75.47%	558.7K

Table 4.4: Comparison of different pooling strategies

Further investigations of the training process leaked the effect that the best training and best validation accuracies are nearly identical, while the training and validation loss, especially for cross-subject validation are diverging. Fig 4.16 visualizes the averaged values of the "Conv Pool" and the "Dense Layer" strategy for the Cross-Subject and the Cross-View accuracy. This leads to the following conclusions:

- Best training and validation accuracy are nearly the same: The model is not able to learn more complex generalization of the input. It indicates an underfitting effect.
- The diverging train and test losses are indicators for the opposite effect: An overfitting.

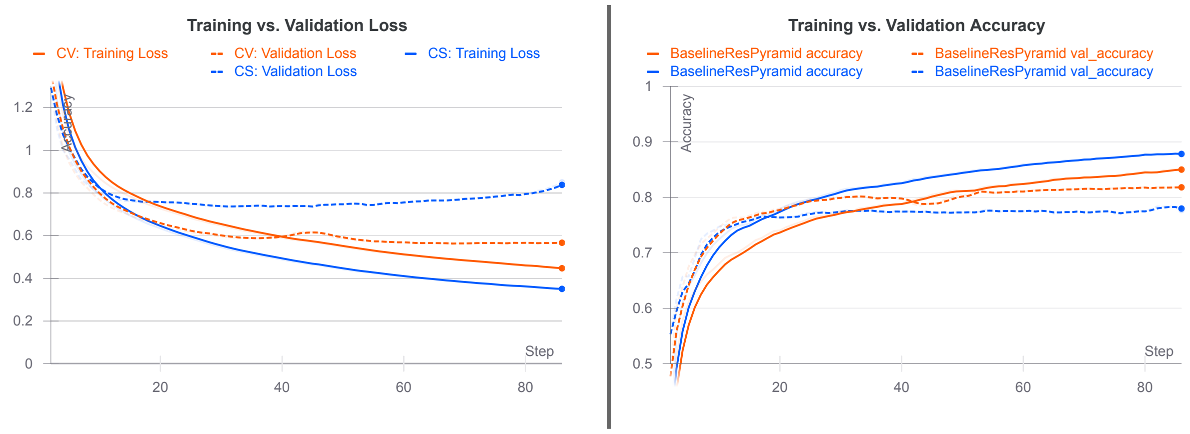


Figure 4.16: Loss and Accuracy of with resolution pyramids extended model, referred as "Original" in table 4.4

Due to the low training accuracy, overfitting does not seem to be probable. The next step would be extending the small network topology depth of the recently used networks by increasing the same. Similar to feature extractors in computer vision networks, convolutions in front of the wide kernels should be added to extract more meaningful features. The goal of these feature extractors is to learn movement primitives analogous to primitive forms like horizontal or vertical edges that are fused to more abstract shapes. To ensure that the wide convolution works as desired, it is crucial to keep the dimension that holds the joint representatives the same size as the input tensor. The results are kernels that convolve single joints over multiple temporal steps. If the convolution would be applied to direct neighbors in x direction, it would lead to incorrect results, because the initial joint order was chosen arbitrarily and has no logical neighborhood relations.

In the new architecture, the input is resized using bilinear interpolation two times, with factor 0.75 and 0.5. Then, three successive convolutional layers that convolve only in temporal dimension are added. The three extractors share their weights over each resolutions pyramid. The same applies to the wide kernels. This time, the training process (fig 4.17) clearly indicates an overfitting. The losses are diverging much more than in the first experiments, while the training accuracy reaches a loss very close to 1 (0.9992). Because the high training accuracy leads to a very small loss, is is impossible to learn additional features.

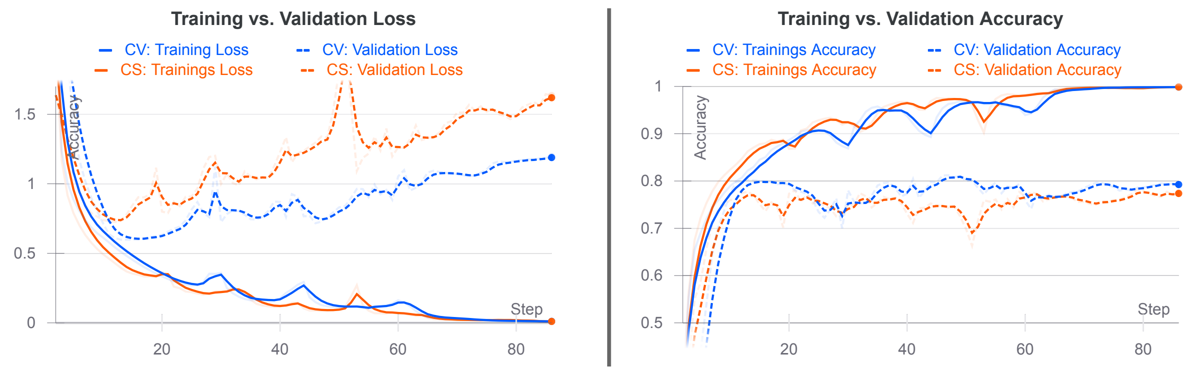


Figure 4.17: Loss and Accuracy of Conv Extraction Variant

In addition, a second variant is developed, that first extracts pose primitives using three stacked convolutional layers followed by the resolution pyramid. The feature maps are then concatenated resolution wise and processed by a dense layer which is finally connected to the softmax output. This second variant was used for the research of preprocessing techniques introduced in section 4.3 "Data Preparation". Both variants are visualized in an abstract form in figure 4.18. Results reveal, that it has no significant effect whether to create pyramids before feature extraction or after (see table 4.5).

Kernel Heights / N Filters	Resolution Pyramid Order	Cross-Subject	Cross-View	FLOPs
[4,6,12], 32	Pre Conv	78.17%	**81.22%**	647.5K
[4,6,12], 32	Post Conv	**78.53%**	80.0%	638.1K

Table 4.5: Comparison Pre-Pyramid and Post-Pyramid Convolution

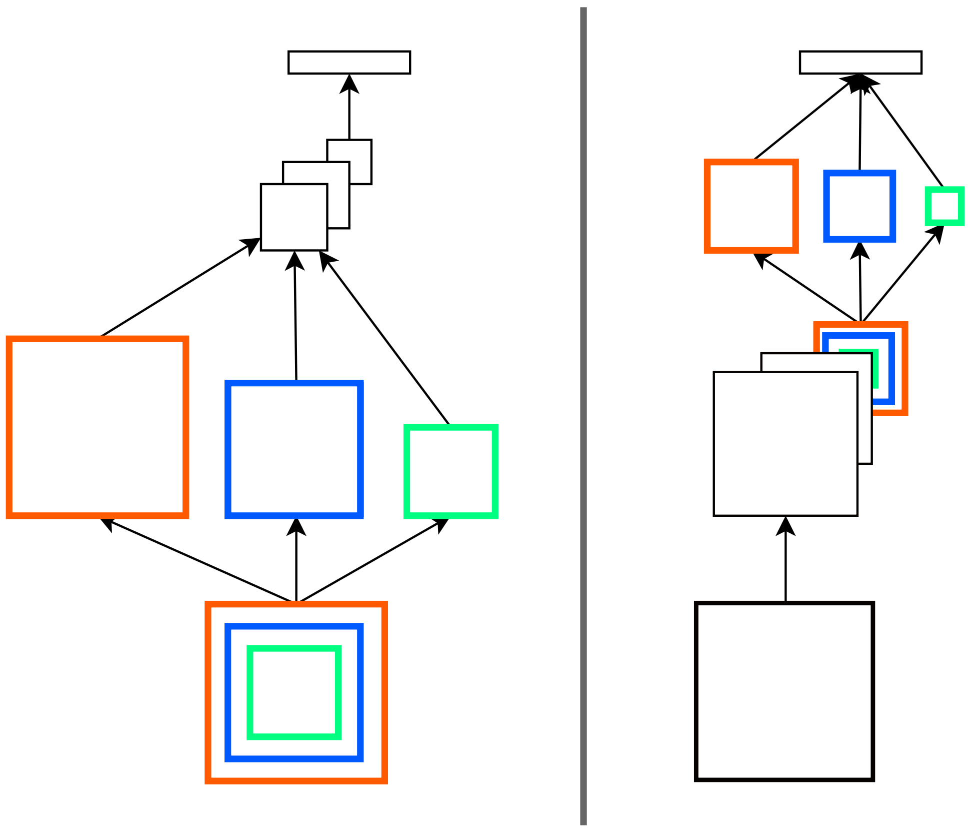


Figure 4.18: Post-Pyramid Convolution (left) vs. Pre-Pyramid Convolution (right). Both reach similar results on the NTU RGB+D dataset (see table 4.5).

4.4.4 Regularization and Rethinking of the Pre-Output Computation

On the one hand, the model capacity was increased, which leads to very high training accuracy close to 1.0 while the validation accuracy could not be improved in the same way. This is a strong indicator for an ovefitting. A network can be prevented from overfitting by regularizing it. Regularization aims at forcing the net to learn more generalized representations of actions instead of perfectly approximating the training dataset.

On the other hand, the concatenation of all max-pooled outputs after the wide convolutions should be changed to make sure that no important temporal information is discarded. In the previous section, multiple pooling techniques were introduced, but none of them improved the accuracy significantly. Therefore, these parts of the network architecture need to be reviewed again.

Regularization

Over- and underfitting are very common issues deep learning researchers need to deal with. A very high training accuracy and a lower validation accuracy in the context of diverging loss values are indicators for such a problem. Best practices for handling these are methods summarized by the term regularization. Widely used techniques are the L2 and the dropout regularization. To prevent the new network from overfitting, an L2 regularization at the wide convolutional layers as well as a dropout layer before the Softmax layer is added.
The L2 regularization penalizes large weights by squaring the parameters of the filters and the bias when the loss is calculated. High weights lead to an even higher loss. This forces the network to learn smaller values which are better balanced instead of very high ones for a single discriminant feature. $5x10^{-4}$ is chosen as a moderate regularization factor. Investigations stated, that a higher regularization prevents the network from learning anything. A lower one could have no effect at all.

The second mentioned technique "dropout layer" randomly "drops" a specified number of features each iteration. This ensures a more uniform distribution of activations per neuron in dense layers. This avoids that the classification is only based on single discriminant features similar to the L2 loss. The chosen dropout factor is 0.5, which means that randomly 50% of activations are rejected before the last layer.

Rethinking Pre-Output Computation

First, max-pooling from the baseline was adapted and one possible issue was found: The temporal information is lost after pooling the whole feature maps into a single value. Multiple pooling options were tested in the previous chapter without a significant performance increase. Therefore, a new mechanism is developed that increases on the one side, the explainability and the understandability of the model and on the other side reaches better validation accuracies. Right after the input layer, three resolution pyramids are built with scale 1, 0.75 and 0.5 followed by three stacked feature extractors sharing weights. Afterward, the modified wide kernels with the additional regularization are added. The main idea of this concept is to extract human pose primitives for each joint separately first. Then, the wide kernels should learn individual combinations of these primitives.

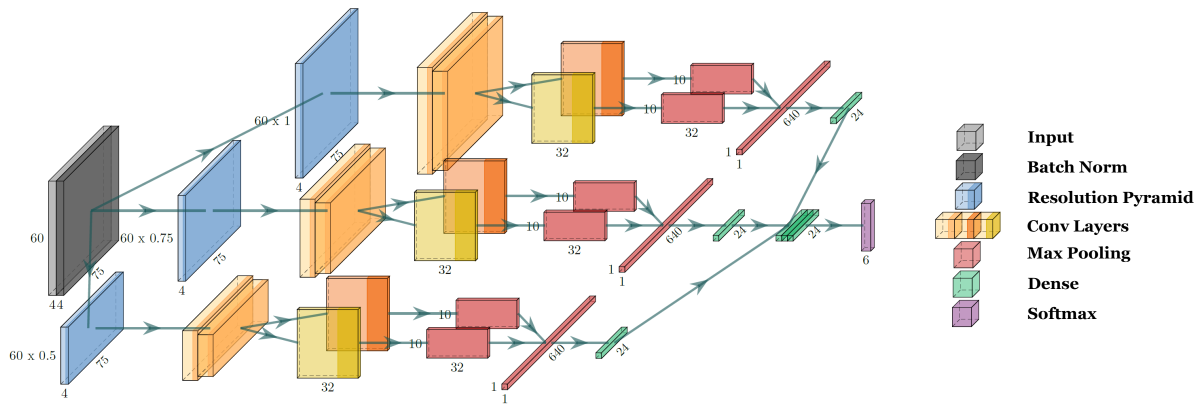


Figure 4.19: The Final IWiCoNet Architecture: Weights are shared in a vertical layers. In Addition, the wide convolutions share the weights with the layers of the other pyramid processed (yellow and dark orange)

The results of each resolution are summarized by downscaling the feature maps to the same size. The maximum activation value at each point is then selected. The compiled maps are discretized to a sequence length of 16 by a max-pooling operations (16 is chosen randomly and could be tuned in the future). Each of the 16 timesteps holds activations of the wide kernels. The connected dense layer then combines the activations at the discretized time steps to a sequence of detected primitive combinations. The dense layer is followed by the final output layer, which finally predicts the action class.
The filter height of the wide kernel are investigated again to reduce the number of parameters. Table 4.6 shows, that reducing the kernel height not necessarily leads to a bad performance.

Kernel Heights / N Filters	Method	Cross-Subject	Cross-View	FLOPs
[4,6,12] / 32	Post Conv	78.83%	81.00%	319.4
[4,6,12] / 32	Latest	80,10%	83.12%	1916.6k
[3,5] / 32	Latest	**82.52%**	**83.02%**	765.9K

Table 4.6: Classification Accuracies of the final *IWiCo-Net*

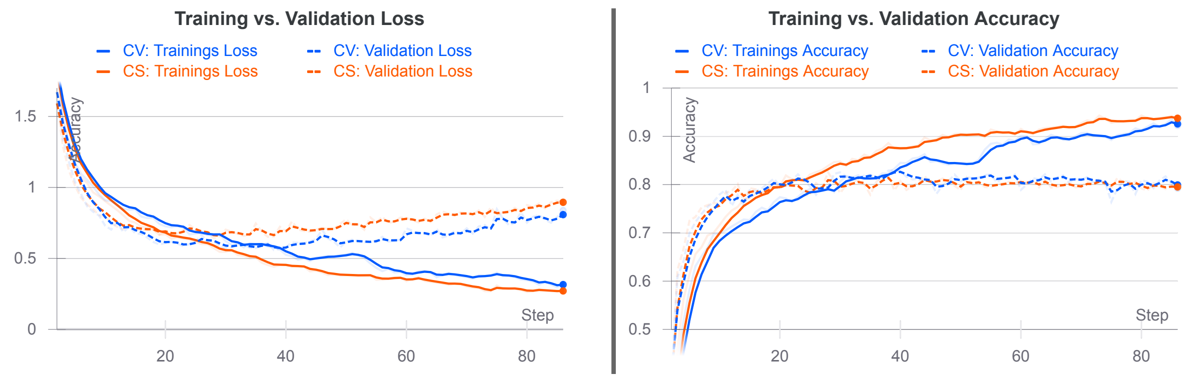


Figure 4.20: Loss and Accuracy of the Final Network

Conclusion

Two improvements were introduced which tackle the problems of overfitting and the extreme information loss. By discretizing the sequences and learning combinations of pose primitives, the overall classification process is better understandable and outperforming all previous techniques, developed in this thesis. It would be a good starting point for additional research. In the following, this architecture is referred as *IWiCo-Net* - Interpretable Wide Convolutions [for skeleton-based action recognizion].

The investigations of different kernel sizes also revealed that the earlier conclusion that several large filters are necessary for good action recognition is no longer valid. This is especially shown in the improved CS score of 82.52% compared to 80.10% with more and bigger filters. The addition of a more advanced feature extraction before wide convolution and the fully connected layer after that results in an increase in FLOPs to 765.9K. Compared to the previous approaches, this is a significant jump. However, if we take the number of parameters of other Action Recognition Nets like the ST-GCN (16.4 GFLOPs) or the current best method according to [?] (36 GFlops), 765K look very slim.

One problem that is not discussed in detail is the significantly worse validation accuracy compared to training accuracy. Although the regularization prevented overfitting on the training data, CS and CV accuracy did not improve as expected. In addition, new methods shrank the CS and CV accuracy distances closer together.

4.4.5 2D Action Recognition in Context

As already stated in section 4.3.5, the NTU dataset does not provide the camera matrices to transform the detected keypoints from 2D into the 3D world coordinate system. Therefore, a transformation is possible, but very time-consuming. Instead of researching the impact to 3D skeleton-based action recognition, the experiments are performed in a two-dimensional space.

Three datasets are prepared for investigations:

- 2D Human Keypoints Only
- 2D Human Keypoints + 17 Object Keypoints (All)
- 2D Human Keypoints + 6 Object Keypoints (TV/Monitor, Laptop, Mouse, Keyboard, Cell Phone, Toothbrush)

The object keypoint fusion technique was introduced in the chapter 2.3 "Recognition Pipeline".

Recap: The most probable interaction object that belongs to a human action/interaction is the closest one. If three smartphones are detected in a scene, the smartphone instance that is the closest one to the subject is chosen as interaction object. The distance between the object and the human is calculated by averaging the sum over the difference vectors magnitudes between the object and the hands, the head, the spine and the feet. This technique can deal with missing joints and ensures a fixed number of joints over all sequences. While the version with 17 keypoints uses all 17 objects listed in chapter 4, the other variant uses only the objects that have no relation with the action "Eating a Meal". With the latter, it is tested whether the overrepresentation of objects relating to class "Eating a Meal" has a negative impact to the other classes.

The previously introduced IWiCo-Net is used for the experiments in combination with the difference vectors between all joints, the spine, the left hand and the right hand. In contrast to the previous section, the human skeletons are not rotated into x,y plane because the 2D coordinate system is used.

The training results of the IWiCo-Net in combination with the three sets are shown in table 4.7. The version using the 6 object classes outperforms the two others.

Kernel Heights / N Filters	Data Type	Cross-Subject	Cross-View	FLOPs
[3,5] / 32	Human Pose Only (HP)	79.79%	82.73%	165.9K
[3,5] / 32	HP + 17 Objects	77.97%	82.51%	1183.7K
[3,5] / 32	HP + 6 Objects	**80.6%**	**84.21%**	912.4K

Table 4.7: 2D Action Recognition: Classification Results

The confusion matrices of the predicted validation sets are visualized in figure 4.21. The left matrix, containing predictions of using human keypoints only, indicates two class overlaps: The class "002 Eating a Meal" is predicted 38 times as "003 Brushing Teeth". Both actions may contain a "Hand to the Mouth" gesture which cannot be distinguished by the skeleton pose alone. The same applies to classes "029 Type on Keyboard" and "030 Play with Phone/Tablet". Both contain sequences of moving hands in a horizontal plane. If the pose would be enriched by interaction objects, which make these actions more distinguishable, the classification result should improve. However, matrices (b) and (c) show that the misclassification of exact these actions still exists. This leads to the conclusion that the enrichment of the human pose by objects has no impact in this experiment.

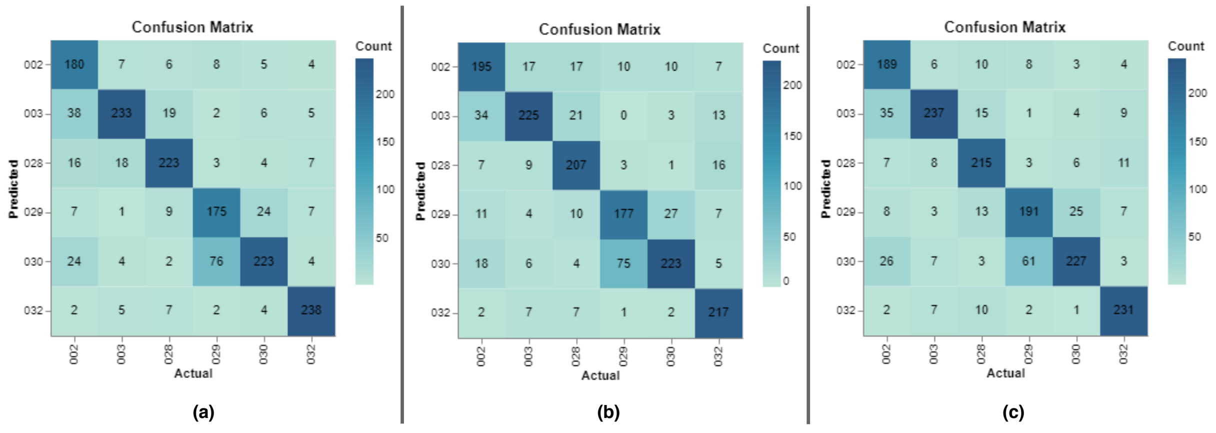


Figure 4.21: Confusion Matrices of the Cross-Subject validation with 2D Data. (a) is the version without additional objects. (b) and (c) using context information

An investigation of the distribution of detected objects in the action sequences indicates the cause for this phenomen: A phone was detected in only 70 percent of the frames of phone action sequence with a mean confidence of 19%. On the other side, in 58% of the three actions without an involved smartphone, a phone was detected with a mean confidence score of 10% percent. These errors exist over all combinations of action classes and interaction objects.

Conclusion

Especially the classes which differ mostly in the used interaction object are hard to distinguish for the neural net. "Eating a Meal" overlaps with "Brushing Teeth" and the same for "Typing on Keyboard" and "Playing with Phone/Tablet". An accurate and reliable detection of interaction objects would help to better distinguish between those hard samples. Unfortunately, the pretrained object detectors are not able to detect small, partially occluded interaction objects properly. Due to this, it is not possible to show an improvement with interaction objects in this work.

5 Conclusion

The main goal of this work was to investigate techniques that can be used for skeleton-based human action recognition in real-world scenarios with a special focus on context knowledge and explainability. Another important aspect is the usage of such algorithms on edge-computing devices to enable these to a wide audience.

With the IWiCo-Net, an architecture was developed which is based on the concept of wide convolutions derived from NLP that can also be applied to human action recognition. The combination of the IWiCo-Net and the introduced data preprocessing techniques leads to acceptable results on a subset of the NTU RGB+D dataset. Approaches like processing the sequences at different scales and the usage of shared weights further increase performance while the complexity is decreased. Thereby, better explainability can be ensured. Nevertheless, the examined approaches can be further improved (see chapter 6 "Outlook").

The investigations of action recognition enriched by context knowledge could not be performed as intended. The missing transformation matrices that enable the transformation of the objects detected in 2D into the 3D coordinate space would only have been possible with an enormous time investment. The workaround was to research the impact of context only in two-dimensional space. Unfortunately, the inclusion of interaction objects did not show any improvement compared to the version using only the human pose. Further analysis of the input data showed that the detection of small and often occluded interaction objects was insufficient and therefore, could be the reason for the unexpected stagnation of accuracy. A closer look at the classification results revealed that the network had problems with those classes that would benefit significantly from the addition of interaction objects.

The IWiCo-Net presented in this thesis benefits from a slim architecture and good adaptability. It is therefore a good base for future practice-oriented research at the Inferics GmbH.

6 Outlook

This work has shown that the concept of wide convolutions can be used for action recognition. To ensure a more reliable recognition of context objects, an improved object detector adapted for this use-case should be implemented. The detector must have a high detection rate for small occluded objects. At the same time, a pose estimator must be chosen to complete the architecture. Both should run fast in an industrial application and at the same time have a sufficiently high reliability.

If the performance is transferable from NTU data to real-world data, the next step should be to investigate how the training can be changed so that the validation accuracy reaches 90% accuracy like for the training set.

The NTU data set itself does not provide sufficient data quality. In particular, missing confidence values and poorly detected poses that are not marked as such may cause noise in the data. A search for a better data set to extend the existing one would be desirable.

Bibliography

[Aubry et al., 2019] Aubry, S., Laraba, S., Tilmanne, J., and Dutoit, T. (2019). Action recognition based on 2D skeletons extracted from RGB videos. *MATEC Web of Conferences*, 277:02034.

[Avola et al., 2019] Avola, D., Cascio, M., Cinque, L., Foresti, G. L., Massaroni, C., and Rodola, E. (2019). 2D Skeleton-Based Action Recognition via Two-Branch Stacked LSTM-RNNs. *IEEE Transactions on Multimedia*, 9210(c):1–1.

[Bagnall et al., 2016] Bagnall, A., Bostrom, A., Large, J., and Lines, J. (2016). The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version.

[Baradel et al., 2018] Baradel, F., Neverova, N., Wolf, C., Mille, J., and Mori, G. (2018). Object level visual reasoning in videos. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11217 LNCS:106–122.

[Barna et al., 2019] Barna, A., Masum, A. K. M., Hossain, M. E., Bahadur, E. H., and Alam, M. S. (2019). A study on Human Activity Recognition Using Gyroscope, Accelerometer, Temperature and Humidity data. *2nd International Conference on Electrical, Computer and Communication Engineering, ECCE 2019*, pages 7–9.

[Bitkom and Deutsches Forschungszentrum für Künstliche Intelligenz, 2017] Bitkom and Deutsches Forschungszentrum für Künstliche Intelligenz (2017). Künstliche Intelligenz: Wirtschaftliche Bedeutung, gesellschaftliche Herausforderungen, menschliche Verantwortung. *Bitkom*, pages 1–208.

[Cao et al., 2016] Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2016). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *Acta Physica Polonica A*, 106(5):709–713.

[Chaaraoui et al., 2013] Chaaraoui, A. A., Padilla-López, J. R., and Flórez-Revuelta, F. (2013). Fusion of skeletal and silhouette-based features for human action recognition with RGB-D devices. *Proceedings of the IEEE International Conference on Computer Vision*, pages 91–97.

[Chakraborty and Talukdar, 2016] Chakraborty, C. and Talukdar, P. (2016). Issues and Limitations of HMM in Speech Processing: A Survey. *International Journal of Computer Applications*, 141(7):13–17.

[Chen et al., 2019] Chen, C., Fragonara, L. Z., and Tsourdos, A. (2019). Go Wider: An Efficient Neural Network for Point Cloud Analysis via Group Convolutions.

[Chen et al., 2015] Chen, C., Jafari, R., and Kehtarnavaz, N. (2015). UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 168–172. IEEE.

[Cho et al., 2015] Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2015). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. pages 103–111.

[Duan et al., 2019] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., and Tian, Q. (2019). CenterNet: Keypoint triplets for object detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:6568–6577.

[Dwarampudi and Reddy, 2019] Dwarampudi, M. and Reddy, N. V. S. (2019). Effects of padding on LSTMs and CNNs.

[Gartner, 2018] Gartner (2018). What Edge Computing Means for Infrastructure and Operations Leaders.

[Gedat et al., 2017] Gedat, E., Fechner, P., Fiebelkorn, R., and Vandenhouten, R. (2017). Human action recognition with hidden Markov models and neural network derived poses. *SISY 2017 - IEEE 15th International Symposium on Intelligent Systems and Informatics, Proceedings*, pages 157–161.

[Glikson and Woolley, 2020] Glikson, E. and Woolley, A. W. (2020). Human Trust in Artificial Intelligence: Review of Empirical Research. *Academy of Management Annals*, page annals.2018.0057.

[He et al., 2017] He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:2980–2988.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

[Jin et al., 2017] Jin, C.-B., Li, S., and Kim, H. (2017). Real-Time Action Detection in Video Surveillance using Sub-Action Descriptor with Multi-CNN. pages 1–29.

[Kabsch, 1976] Kabsch, W. (1976). A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923.

[Kalchbrenner et al., 2014] Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, 1:655–665.

[Kim et al., 2019] Kim, S., Yun, K., Park, J., and Choi, J. Y. (2019). Skeleton-based action recognition of people handling objects. *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, pages 61–70.

[Kim, 2014] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1746–1751.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2:1097–1105.

[Li et al., 2017a] Li, C., Hou, Y., Wang, P., and Li, W. (2017a). Joint Distance Maps Based Action Recognition With Convolutional Neural Networks. *IEEE Signal Processing Letters*, 24(5):624–628.

[Li et al., 2017b] Li, C., Zhong, Q., Xie, D., and Pu, S. (2017b). Skeleton-based Action Recognition with Convolutional Neural Networks. *IEEE Signal Processing Letters*, 24(5):624–628.

[Li et al., 2017c] Li, W., Wen, L., Chang, M.-C., Lim, S. N., and Lyu, S. (2017c). Adaptive RNN Tree for Large-Scale Human Action Recognition. In *2017 IEEE International Conference on Computer Vision (ICCV)*, volume 2017-Octob, pages 1453–1461. IEEE.

[Li et al., 2017d] Li, Y., Ouyang, W., Wang, X., and Tang, X. (2017d). ViP-CNN: Visual phrase guided convolutional neural network. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:7244–7253.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327.

[Masum et al., 2019] Masum, A. K. M., Bahadur, E. H., Shan-A-Alahi, A., Uz Zaman Chowdhury, M. A., Uddin, M. R., and Al Noman, A. (2019). Human Activity Recognition Using Accelerometer, Gyroscope and Magnetometer Sensors: Deep Neural Network Approaches.

2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019, pages 6–11.

[Rabiner and Juang, 1986] Rabiner, L. R. and Juang, B. H. (1986). An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, 3(1):4–16.

[Reddi et al., 2019] Reddi, S. J., Kale, S., and Kumar, S. (2019). On the Convergence of Adam and Beyond. pages 1–23.

[Ren et al., 2020] Ren, B., Liu, M., Ding, R., and Liu, H. (2020). A Survey on 3D Skeleton-Based Action Recognition Using Learning Method. pages 1–8.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

[Sempena et al., 2011] Sempena, S., Nur Ulfa Maulidevi, and Peb Ruswono Aryan (2011). Human action recognition using Dynamic Time Warping. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, number July, pages 1–5. IEEE.

[Senin, 2008] Senin, P. (2008). Dynamic Time Warping Algorithm Review. *Science*, 2007(December):1–23.

[Shahroudy et al., 2016] Shahroudy, A., Liu, J., Ng, T. T., and Wang, G. (2016). NTU RGB+D: A large scale dataset for 3D human activity analysis. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:1010–1019.

[Shi et al., 2019] Shi, L., Zhang, Y., Cheng, J., and Lu, H. (2019). Skeleton-Based Action Recognition with Multi-Stream Adaptive Graph Convolutional Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:12018–12027.

[Strategy Analytics, 2020] Strategy Analytics (2020). New Record for Smart Speakers As Global Sales Reached 146.9 Million in 2019.

[Wang et al., 2016] Wang, P., Li, Z., Hou, Y., and Li, W. (2016). Action recognition based on joint trajectory maps using Convolutional Neural Networks. *MM 2016 - Proceedings of the 2016 ACM Multimedia Conference*, pages 102–106.

[Yan et al., 2018] Yan, S., Xiong, Y., and Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 7444–7452.

[Zanfir et al., 2013] Zanfir, M., Leordeanu, M., and Sminchisescu, C. (2013). The Moving Pose: An Efficient 3D Kinematics Descriptor for Low-Latency Action Recognition and Detection. In *2013 IEEE International Conference on Computer Vision*, pages 2752–2759. IEEE.

[Zhang et al., 2019] Zhang, H. B., Zhang, Y. X., Zhong, B., Lei, Q., Yang, L., Du, J. X., and Chen, D. S. (2019). A comprehensive survey of vision-based human action recognition methods. *Sensors (Switzerland)*, 19(5):1–20.

[Zhang et al., 2018] Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., and Zheng, N. (2018). View Adaptive Neural Networks for High Performance Skeleton-based Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1963–1978.

[Zhang et al., 2017] Zhang, S., Liu, X., and Xiao, J. (2017). On geometric features for skeleton-based action recognition using multilayer LSTM networks. *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, pages 148–157.

[Zhang and Wallace, 2015] Zhang, Y. and Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.

[Zhu et al., 2018] Zhu, J., Zou, W., Xu, L., Hu, Y., Zhu, Z., Chang, M., Huang, J., Huang, G., and Du, D. (2018). Action Machine: Rethinking Action Recognition in Trimmed Videos.