

Deep Learning for Hand Gesture Recognition on Skeletal Data

Guillaume Devineau, Wang Xi, Fabien Moutarde, Jie Yang

▶ **To cite this version:**

Guillaume Devineau, Wang Xi, Fabien Moutarde, Jie Yang. Deep Learning for Hand Gesture Recognition on Skeletal Data. 13th IEEE Conference on Automatic Face and Gesture Recognition (FG'2018), May 2018, Xi'An, China. 10.1109/FG.2018.00025 . hal-01737771

HAL Id: hal-01737771

https://minesparis-psl.hal.science/hal-01737771

Submitted on 22 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Deep Learning for Hand Gesture Recognition on Skeletal Data

Guillaume Devineau[1] and Wang Xi[2] and Fabien Moutarde[1] and Jie Yang[2]
[1] MINES ParisTech, PSL Research University, Center for Robotics, 60 Bd St Michel 75006 Paris, France
[2] Shanghai Jiao Tong University, School of Electronic Information and Electrical Engineering, Shangai, China

***Abstract*— In this paper, we introduce a new 3D hand gesture recognition approach based on a deep learning model.**

We introduce a new Convolutional Neural Network (CNN) where sequences of hand-skeletal joints' positions are processed by parallel convolutions; we then investigate the performance of this model on hand gesture sequence classification tasks. Our model only uses hand-skeletal data and no depth image.

Experimental results show that our approach achieves a state-of-the-art performance on a challenging dataset (DHG dataset from the SHREC 2017 3D Shape Retrieval Contest), when compared to other published approaches. Our model achieves a 91.28% classification accuracy for the 14 gesture classes case and an 84.35% classification accuracy for the 28 gesture classes case.

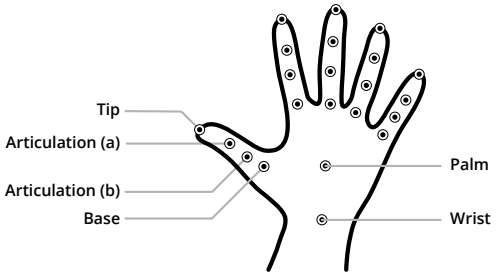


Fig. 1. Hand skeleton returned by the Intel RealSense camera. Each dot represents one of the $n = 22$ joints of the skeleton.

I. Introduction

Touch and gesture are two natural ways for a user to interact with one's environment. While touch necessarily involves a physical contact (e.g. to write a message on a phone, to grab a physical object, or to swipe touch-sensitive textiles), gestures allow remote interactions (e.g. to interact with a smart screen, or with virtual-reality and augmented-reality objects). As such, gesture-based human-computer interfaces can ease the use of digital computing [27] in situations where it would previously have been difficult or even impossible because of practical constraints like interacting with everyday life physical objects (e.g. lights, mirrors, doorknobs, notebooks, ...) or like using computers in settings where the person has to focus entirely on a task (e.g. while driving a car, cooking or doing surgery).

Gesture can convey semantic meaning, as well as contextual information such as personality, emotion or attitude. For instance, research shows that speech and gesture share the same communication system [2] and that one's gestures are directly linked to one's memory [18]. Among gestures, hand gestures distinguish themselves from two other types of gestures [25]: body gestures and head gestures. We chose to work on hand gestures since they can carry more information more easily than the two other types of gestures. One preferred way to infer the intent of a gesture is to use a taxonomy of gestures and to classify the unknown gesture into one of the existing categories based of the gesture data, in a similar way to what is done in computer vision for instance. The classification can either be obtained in realtime at each time step or at the end of the gesture, depending on the the processing power and the application needs.

In this paper we propose a convolutional neural network architecture relying on intra- and inter- parallel processing of sequences of positions (of hand-skeletal joints) to classify complete hand gestures. Where most existing deep learning approaches to gesture recognition use RGB-D image sequences to classify gestures [49], our neural network only uses hand (3D) skeletal data sequences which are quicker to process than image sequences.

The rest of this paper is structured as follows. We first review common recognition methods in Section II. We then present the DHG dataset we used to evaluate our network in Section III. We detail our approach in Section IV in terms of motivations, architecture and results. Finally, we conclude in Section VI and discuss how our model can be improved and integrated into a realtime interactive system.

II. Definition & Related Work

We define a 3D skeletal data sequence s as a vector

$$s = (p_1 \cdots p_n)^T$$

whose components p_i are multivariate time sequences. Each component $p_i = (p_i(t))_{t\in\mathbb{R}}$ represents a multivariate sequence with three (univariate sequences) components

$$p_i = (x^{(i)}, y^{(i)}, z^{(i)})$$

that alltogether represent a time sequence of the positions $p_i(t)$ of the i-th skeletal joint j_i. Every skeletal joint j_i represents a distinct and precise articulation or part of one's hand in the physical world. An illustration of a 3D hand skeleton is proposed in figure 1.

In the following subsections, we present a short review of some approaches to gesture recognition. Typical approaches to hand gesture recognition begin with the extraction of spatial and temporal features from raw data. The features are later classified by a Machine Learning algorithm. The

feature extraction step can either be explicit, using hand-crafted features known to be useful for classification, or implicit, using (machine) learned features that describe the data without requiring human labor or expert knowledge. Deep Learning algorithms leverage such learned features to obtain hierarchical representations (features) that often describe the data better than hand-crafted features. As we work on skeletal data only, with a deep-learning perspective, this review pays limited attention to non deep-learning based approaches and to depth-based approaches; a survey on the former approaches can be found in [25] while several recent surveys on the latter approaches are listed in NEVEROVA's thesis [28].

A. Non-deep-learning methods using hand-crafted features

Various hand-crafted representations of skeletal data can be used for classification. These representations often describe physical attributes and constraints, or easily interpretable properties and correlations of the data, with an emphasis on geometric features and statistical features. Some commonly used features are the position of the skeletal joints, the orientation of the joints, the distance between joints, the angles between joints, the curvature of the joints' trajectories, the presence of symmetries in the skeletal, and more generally other features that involve a human interpretable metric calculated from the skeletal data [21], [22], [41]. For instance, in [45], VEMULAPALLI *et al.* propose a human skeletal representation that lies in the Lie group $SE(3) \times \ldots \times SE(3)$, based on the idea that rigid body rotations and translations in 3D space are members of the Special Euclidean group $SE(3)$. Human actions are then viewed as curves in this curved manifold. Recognition (classification) is finally performed in the corresponding Lie algebra. In [11], DEVANNE *et al.* represent skeletal joints' sequences as trajectories in a n-dimensional space; the trajectories of the joints are then interpreted in a Riemannian manifold. Similarities between the shape of trajectories in this shape space are then calculated with k-Nearest Neighbor (k-NN) to achieve the sequence classification. In [9], two approaches for gesture recognition -on the DHG dataset presented in the next section- are presented. The first one, proposed by GUERRY *et al.*, is a deep-learning method presented in the next subsection. The second one, proposed by DE SMEDT *et al.*, uses three hand-crafted descriptors: Shape of Connected Joints (SoCJ), Histogram of Hand Directions (HoHD) and Histogram of Wrist Rotations (HoWR), as well as Fisher Vectors (FV) for the final representation.

Regardless of the features used, hand-crafted features are always fed to a classifier to perform the gesture recognition. In [7], CIPPITELLI *et al.* use a multi-class Support Vector Machine (SVM) for the final classification of activity features based on posture features. Other very frequently used classifiers [48] are the Dynamic Time Warping (DTW) algorithm, discrete distance-based methods, Naive Bayes, Hidden Markov models (HMM), Conditional Random Fields (CRF) and even simple k-Nearest Neighbors (k-NN).

B. Deep-Learning based methods

Deep Learning, also known as Hierarchical Learning, is a subclass of Machine Learning where algorithms f use a cascade of non-linear computational units f_i (layers) for feature extraction and transformation: $f = f_1 \circ f_2 \circ \ldots \circ f_n$ [10]. The composition of (learned) functions in deep learning algorithms allows them to build a hierarchical representation of the data; that representation can become more and more abstract as the number of layers in the composition goes up. Since deep learning classification models both learn the features to use for the classification and also the mapping from the feature space to the output classes, they reveal themselves to be a very convenient class of models for classification. Moreover, in the recent years deep learning approaches have led to state-of-the-art results for numerous tasks across various domains such as Speech Recognition, Image Recognition or Natural Language Processing [13].

A traditional CNN model almost always involves a sequence of convolution and pooling layers, that are followed by hidden dense layers. Convolution and pooling layers serve as feature extractors, whereas the hidden layers, also called Multi Layer Perceptron (MLP), can be seen as a classifier.

A strategy to mix deep-learning algorithms and (hand) gesture recognition consists in training Convolutional Neural Networks (CNNs, ConvNets) [24] on RGB-D images.

A common trick to quickly train CNNs on gesture datasets consists in doing Transfer Learning [34]. Transfer Learning can be described as the extraction of knowledge from a learning task, knowledge that is later leveraged to help improve the learning of a different, but related, task. In the case of CNNs, transfer learning typically consists in using well performing CNNs already trained for image classification tasks on datasets such as ImageNet [35] and retraining the last layers of these CNNs on a gesture dataset, with all the other layers' weights frozen. To be fine-tuned, the CNN is then retrained one more time on the gesture dataset, with no weight frozen this time. Transfer Learning can also be used for other gesture recognition goals, such as extending gesture vocabularies [16], [14].

A direct example of hand gesture recognition via image CNNs can be found in the works of STREZOSKI *et al.* [40] where CNNs are simply applied on the RGB images of sequences to classify. GUERRY *et al.* [9] propose a deep-learning approach for hand gesture recognition on the DHG dataset, which is described at a later stage of this paper. The GUERRY *et al.* approach consists in concatenating the Red, Green, Blue and Depth channels of each RGB-D image. An already pretrained VGG [37] image classification model is then applied on sequences of 5 concatenated images consecutive in time.

In [26], MOLCHANOV *et al.* introduce a CNN architecture for RGB-D images where the classifier is made of two CNN networks (a high-resolution network and a low-resolution network) whose class-membership outputs are fused with an element-wise multiplication.

NEVEROVA *et al.* carry out a gesture classification task on multi-modal data (RGB-D images, audio streams and

skeletal data) in [29], [30]. Each modality is processed independently with convolution layers at first, and then merged. To avoid meaningless co-adaptation of modalities a multi-modal dropout (ModDrop) is introduced.

Nevertheless, these approaches use depth information where we only want to use skeletal data.

In [46], WANG *et al.* color-code the joints of a 3D skeleton across time. The colored (3D) trajectories are projected on 2D planes in order to obtain images that serve as inputs of CNNs. Each CNN emits a gesture class-membership probability. Finally, a class score (probability) is obtained by the fusion of the CNNs scores.

Recurrent Neural Networks (RNN), e.g. networks that use Long Short-Term Memory (LSTM) [19] or Gated Recurrent Units (GRU) [6], have long been considered as the best way to achieve state-of-the-art results when working with neural networks on sequences like time series. Recently, the emergence of new neural networks architectures that use convolutions or attention mechanisms [43], [44] rather than recurrent cells has challenged this assumption, given that RNNs present some significant issues such as being sensitive to the first examples seen, having complex dynamics that can lead to chaotic behavior [23] or being models that are intrinsically sequential, which means that their internal state computations are difficult to parallelize, to name only a few of their issues.

In [38], SONG *et al.* elegantly combine the use of an LSTM-based neural network for human action recognition from skeleton data with a spatio-temporal attention mechanism. While this approach seems promising, the current paper rather seeks to find a convolution-only architecture rather than a recurrent one.

ZHENG *et al.* propose a convolution-based architecture that does not involve recurrent cells in [50], although this architecture can easily be extended with recurrent cells: [32]. ZHENG *et al.* introduce a general framework (Multi-Channels Deep Convolution Neural Networks, or MC-DCNN) for multivariate sequences classification. In MC-DCNN, multivariate time series are seen as multiple univariate time series; as such, the neural network input consist of several 1D time series sequences. The feature learning step is executed on every univariate sequence individually. The respective learned features are later concatenated and merged using a classic MLP placed at the end of the feature extraction layers to perform classification. The major difference between MC-DCNN and other deep (skeletal) gesture recognition models lies in the fact that MC-DCNN networks are skeleton-structure agnostic. A naive direct use of the model proposed by that paper does nevertheless not yield to results significantly competitive against other approaches results, but still gives a first glimpse of neural architectures for multivariate sequences such as hand gesture skeleton data. The current paper introduces a new neural network built upon this framework.

III. DATASET

To evaluate several variations of the presented neural network model architecture and their performances we conducted experiments on the Dynamic Hand Gesture-14/28 (DHG) dataset [9] created and introduced by DE SMEDT *et al.* in the SHREC2017 - 3D Shape Retrieval Contest.

The DHG dataset consists in a total of 2800 labeled hand gesture sequences performed by 28 participants. The sequences are recorded by an Intel RealSense depth camera and have variable lengths. Each labeled sequence consists of the raw data sequence returned by the camera, associated with two labels representing the category of the recorded gesture. For all sequences a depth image of the scene is provided at each timestep, alongside with both a 2D and a 3D skeletal representation of the hand. The hand skeleton returned by the Intel RealSense depth camera is presented in a paragraph below. Each gesture falls into one of 14 categories : `Grab (G)`, `Tap (T)`, `Expand (E)`, `Pinch (P)`, `Rotation clockwise (RC)`, `Rotation counter-clockwise (RCC)`, `Swipe right (SR)`, `Swipe left (SL)`, `Swipe up (SU)`, `Swipe down (SD)`, `Swipe x (SX)`, `Swipe + (S+)`, `Swipe v (SV)`, `Shake (Sh)`. Moreover, each gesture can be performed with either only one finger or with the whole hand. That means that gestures are classified with either 14 labels or 28 labels, depending on the number of fingers used.

The gesture recognition method we introduce in the next section only uses the 3D hand skeletal representation returned by the Intel RealSense depth camera. At each time step the 3D hand skeleton consists in an ordered list of 22 joints with their positions $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3, \ \forall i \in [\![1;22]\!]$ in the 3D space. The structure of the skeletal returned by the camera is presented in figure 1.

The dataset is split into 1960 train sequences (70% of the dataset) and 840 test sequences (30% test sequences).

IV. PARALLEL CONVOLUTIONS MODEL

A. Motivation

The goals of the original contest where the DHG dataset was introduced were to (1) study the dynamic hand gesture recognition using depth and full hand skeleton, and to (2) evaluate the effectiveness of recognition process in terms of coverage of the hand shape that depend on the number of fingers used. Nevertheless, the goals of this paper are different. Its first goal is to demonstrate that carrying out hand gesture recognition with a sparse representation of the hand (i.e. the 3D hand skeleton) only is competitive with other existing approaches that often focus on RGB-D images. The second goal of this paper is to propose a generic neural network that does not require recurrent cells to achieve this recognition.

B. Model Architecture

We introduce a multi-channel convolutional neural network with two feature extraction modules and a residual branch per channel. The whole model architecture is depicted in figure 2. The architecture is inspired by the high-resolution and low-resolution networks from [26]. The use of residual branches in our architecture is inspired from the

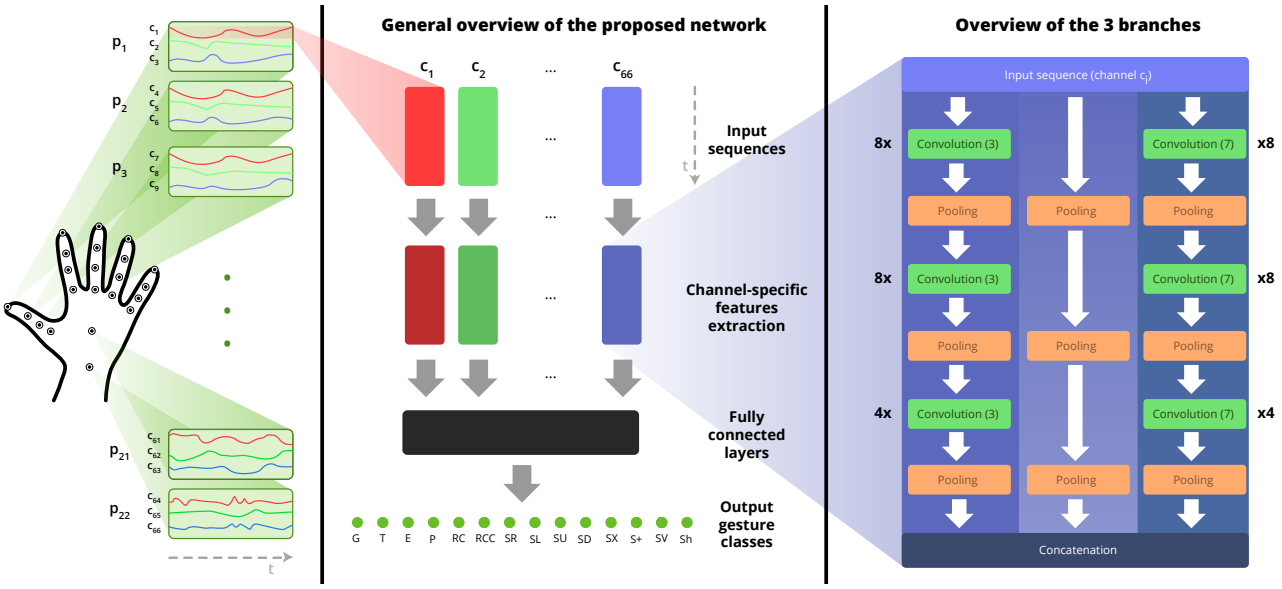


Fig. 2. Illustration of the proposed parallel convolutional neural network. Every channel is processed separately before the Multi Layer Perceptron. The parallel feature extraction module presented on the right is not shared between the 66 channels.

original Residual Networks paper [17]. Residual branches make networks easier to optimize because of a better gradient backpropagation in the training phase; they empirically improve the accuracy of deep networks.

Our network inputs consist of multiple, fixed-length, 1D sequences $(s_1, s_2, \ldots, s_c)$ where $c \in \mathbb{N}^*$ is the number of sequences (channels). Each of these sequences s_i is directly fed to three parallel branches. The first branch[1], improperly called residual branch in this paper, is almost an identity function. Instead of outputting exactly its input we perform a pooling on the input in order to reduce the risk of overfitting. The second and third branches both present a similar architecture, detailed below, designed for feature extraction.

In these two branches, the input is processed as follows. The input is passed to a convolution layer, whose output is subsampled using a pooling layer. This process is repeated two more times. For a single branch, the difference between all the three convolutions resides in the number of feature maps used; the difference between the two branches resides in the size of the convolutions kernels. Having two kernel sizes for the time convolution layers allows the network to directly work at different time resolutions.

Formally, let $h^{(l,\beta)}$ represent the input of the l-th convolution layer of the β branch, $K^{(l,\beta)}$ be the number of feature maps, $W_k^{(l,\beta)}$ the k-th convolution filter of the l-th convolution in the β branch, and $b_k^{(l,\beta)}$ the bias shared for the k-th filter map. The output $h^{(l+1,\beta)}$ of the l-th convolution layer is calculated as

$$h^{(l+1,\beta)} = \sigma\left(h^{(l,\beta)} * W_k^{(l,\beta)} + b_k^{(l,\beta)}\right)$$

where σ is the activation function. This output $h^{(l+1,\beta)}$ serves as the input of the pooling layer that directly follows the convolution layer.

For a single channel, the outputs of the three branches are concatenated into a single vector. Finally, a multi layer perceptron "merges" the -concatenated- vectors of all the channels together and acts as a classifier. There are as many MLP outputs as the number of gesture classes.

In our experiments, we have two branches (high resolution and low resolution branches): $\beta \in [\![1;2]\!]$, 3 convolution and pooling layers: $l \in [\![1;3]\!]$, and $K^{(l,\beta)} = 8$ feature maps for $l = 1$ or $l = 2$ and $K^{(l,\beta)} = 4$ feature maps for $l = 3$. The multi layer perceptron has 1 hidden layer with 1996 hidden units. All of the neurons in our network use the ReLU activation function: $\sigma(x) = \mathrm{ReLU}(x) = \max(0, x)$, with the exception of the output neurons which use the softmax activation function. All of the 3 $[\times 2 \times c]$ subsampling layers use an average pooling with a temporal pool size of 2. Average pooling computes the average value of features in a neighborhood (of 2 time steps in our case), while max pooling extracts the maximum value of the features in the neighborhood. Empirically, it has been shown that max pooling outperforms average pooling in image recognition problems [3]. Nevertheless, experiments we conducted on the choice of the pooling method for this model did not conclude that our model exhibits better results with max pooling (we see a 0.88% decrease in validation accuracy for the model with maximum pooling rather than the average one for the model configuration presented in this paper).

C. Training

In this section we detail the hyperparameters we used as well as some other information related to the training.

1) Data preprocessing: Multidimensional full sequences of joints' trajectories were split into unidimensional sequences that were later re-sampled in order to serve as inputs of our model. The resize was performed by a simple linear

[1]middle branch in figure 2

	G(1)	G(2)	T(1)	T(2)	E(1)	E(2)	P(1)	P(2)	RC(1)	RC(2)	RCC(1)	RCC(2)	SR(1)	SR(2)	SL(1)	SL(2)	SU(1)	SU(2)	SD(1)	SD(2)	SX(1)	SX(2)	S+(1)	S+(2)	SV(1)	SV(2)	Sh(1)	Sh(2)
G(1)	71.4	17.9	0.0	3.6	0.0	0.0	0.0	0.0	0.0	0.0	7.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
G(2)	0.0	96.7	0.0	0.0	0.0	0.0	0.0	3.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T(1)	15.2	0.0	54.5	15.2	0.0	0.0	6.1	0.0	3.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0
T(2)	0.0	10.7	0.0	57.1	0.0	3.6	0.0	0.0	0.0	10.7	0.0	7.1	0.0	0.0	0.0	0.0	0.0	3.6	0.0	0.0	0.0	0.0	0.0	3.6	3.6	0.0	0.0	0.0
E(1)	0.0	0.0	0.0	0.0	85.2	3.7	3.7	0.0	0.0	7.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
E(2)	0.0	0.0	0.0	3.6	10.7	71.4	0.0	3.6	0.0	3.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.6
P(1)	7.4	0.0	0.0	0.0	0.0	0.0	85.2	3.7	0.0	0.0	3.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P(2)	0.0	8.3	0.0	0.0	0.0	0.0	8.3	75.0	0.0	8.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RC(1)	0.0	0.0	3.4	0.0	0.0	0.0	0.0	0.0	69.0	24.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.4	0.0
RC(2)	0.0	0.0	0.0	3.8	0.0	0.0	0.0	0.0	15.4	80.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RCC(1)	0.0	0.0	3.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	81.3	15.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RCC(2)	0.0	15.4	3.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.7	69.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.8	0.0
SR(1)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	87.9	9.1	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SR(2)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	17.2	79.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.4	0.0	0.0	0.0	0.0
SL(1)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.8	0.0	0.0	0.0	0.0	0.0	84.6	7.7	0.0	0.0	0.0	3.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SL(2)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.1	0.0	0.0	0.0	0.0	7.1	85.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SU(1)	0.0	0.0	0.0	3.1	12.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	78.1	3.1	0.0	0.0	0.0	0.0	0.0	0.0	3.1	0.0	0.0	0.0
SU(2)	0.0	0.0	0.0	0.0	0.0	13.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	13.9	69.4	0.0	2.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SD(1)	0.0	0.0	0.0	3.2	0.0	0.0	6.5	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	80.6	3.2	0.0	0.0	0.0	0.0	3.2	0.0	0.0	0.0
SD(2)	0.0	13.3	0.0	3.3	0.0	0.0	0.0	0.0	0.0	3.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.3	0.0	73.3	0.0	0.0	0.0	0.0	0.0	3.3	0.0	0.0
SX(1)	0.0	0.0	0.0	3.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	90.9	0.0	0.0	0.0	3.0	0.0	0.0	0.0
SX(2)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	97.2	0.0	2.8	0.0	0.0	0.0	0.0
S+(1)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0
S+(2)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0
SV(1)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	89.3	7.1	0.0	0.0
SV(2)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	96.7	0.0	0.0
Sh(1)	0.0	0.0	2.9	0.0	0.0	0.0	8.6	0.0	11.4	5.7	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	60.0	8.6
Sh(2)	0.0	0.0	0.0	0.0	0.0	2.9	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	91.4

In this confusion matrix each row represents the real class of performed gestures while each column represents the predicted class of the gestures.

TABLE I

CONFUSION MATRIX FOR DHG-28 USING OUR PROPOSED APPROACH

interpolation of the time series. The few data points outside the boundaries of the input were filled with a reflection. After interpolation, every sequence has a fixed length of 100 time steps.

2) Weights Initialization & Batching: Each training batch contained a set of 32 skeletal gesture sequences of length 100, where a skeletal gesture sequence is a list of $22 \times 3 = 66$ unidimensional sequences.

For the training, we used the Xavier initialization (also known as GLOROT uniform initialization) [12] to set the initial random weights for all the weights of our model.

3) Implementation: Our model was implemented twice using either `PyTorch` or `Keras` as an high level library above `Tensorflow`. `CUDA`/`CuDNN` were used for GPU parallelization of the computations.

4) Hardware: We trained our model on one machine with a GPU (NVIDIA GeForce GTX 1080 Ti). Using the hyperparameters presented in the paper, each training step took about 12 seconds. We trained the model for 1000 steps. As a comparison, other experiments -not presented in this paper- that involved GRU recurrent cells were trained for over 10000 steps of approximately 45s each while not exhibiting better performance on the testing data.

5) Loss & Optimizer: We selected negative log-likelihood as the cost function. To train our model, we used the popular Adam optimization algorithm [20] which calculates an exponential moving average of the gradient and the squared gradient. For the decay rates of the moving averages we used the parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$. The values of other parameters were $\alpha = 10^{-3}$ for the learning rate, and $\varepsilon = 10^{-8}$.

6) Regularization: Dropout [39] served as a regularizer, with a drop rate of $p = 0.2$ for the model presented in this paper. Numerous experiments on variations of both the model architecture and the dropout rate empirically showed that higher dropout rates like $p = 0.4$ often reduced the training accuracy but did not increase the testing accuracy.

D. Results on the DHG dataset

We work on the DHG dataset presented on section III. All sequences are preprocessed as described in the previous section. Each resampled skeletal sequence is split into 22 joints' sequences, and then into 3 $\times$ 1D sequences of the (x,y,z) positions of the joints. This leads to $66 = 22 \times 3$ input sequences that are fed to model we introduced. The outputs of the last layer of the MLP represent the labels we will attribute to the gesture corresponding to the 66-channel input. Since there are two classification tasks, depending on the number of classes, we use two different neural networks with the same architecture, except that one has 14 outputs and the other has 28 outputs.

On the DHG dataset, our model achieves a 91.28% classification accuracy for the 14 gesture classes case and an 84.35% classification accuracy for the 28 gesture classes case. These are the best recognition accuracy scores known for this challenging dataset at this day.

In [33], [11], [9], [31], [8] more or less complex but handcrafted features are used in the classification pipelines. The main advantage of deep-learning approaches is to automatically discover such (sometimes complex) features. [5] is based on deep-learning architecture. It directly applies LSTMs -but without applying CNNs beforehand- to the skeletal data (and to the handcrafted features). Introducing a convolution step before the LSTMs could possibly improve the model in [5]. Our model likely uses more efficient representations due to the use of the parallel branches. A comparison of the different approaches is presented in table II.

In classification, precision is defined as the ratio $\text{precision} = \frac{TP}{TP+FP}$ where TP is the number of true positives

Approaches	Accuracy 14 gestures	Accuracy 28 gestures
OREIFEJ & LIU [33]	78.53	74.03
DEVANNE *et al.* [11]	79.61	62.00
GUERRY *et al.* [9]	82.90	71.90
OHN-BAR & TRIVEDI [31]	83.85	76.53
CHEN *et al.* [5]	84.68	80.32
DE SMEDT *et al.* [8]	88.24	81.90
Ours	**91.28**	**84.35**

TABLE II

ACCURACY RESULTS ON THE DHG DATASET

	G	T	E	P	RC	RCC	SR	SL	SU	SD	SX	S+	SV	Sh
G	94.8	1.7	0.0	1.7	1.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
T	11.5	77.0	0.0	3.3	4.9	0.0	0.0	0.0	0.0	3.3	0.0	0.0	0.0	0.0
E	0.0	5.5	90.9	0.0	0.0	0.0	1.8	0.0	0.0	0.0	0.0	0.0	0.0	1.8
P	15.7	0.0	0.0	78.4	3.9	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RC	0.0	1.8	0.0	0.0	98.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RCC	3.4	6.9	0.0	1.7	5.2	77.6	0.0	1.7	0.0	1.7	0.0	0.0	0.0	1.7
SR	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SL	0.0	0.0	0.0	0.0	7.4	0.0	3.7	88.9	0.0	0.0	0.0	0.0	0.0	0.0
SU	2.9	2.9	10.3	0.0	0.0	0.0	1.5	0.0	79.4	1.5	0.0	0.0	1.5	0.0
SD	3.3	0.0	0.0	0.0	3.3	0.0	0.0	0.0	0.0	91.8	0.0	0.0	1.6	0.0
SX	0.0	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	89.9	1.4	5.8	0.0
S+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
SV	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0
Sh	0.0	8.6	2.9	0.0	12.9	0.0	2.9	0.0	1.4	0.0	0.0	0.0	0.0	71.4

In this confusion matrix each row represents the real class of performed gestures while each column represents the predicted class of the gestures.

TABLE III

CONFUSION MATRIX FOR DHG-14 USING OUR PROPOSED APPROACH

and FP the number of false positives, while recall is defined as the ratio recall $= \frac{TP}{TP+FN}$ where TP still is the number of true positives and FN is the number of false negatives. The F_1 score (given by $F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$) can be interpreted as a weighted average of the precision and recall; an F_1 score reaches its best value at 1 and worst score at 0. A comparison of F_1 scores between our approach and the best performing method that exists at this day shows that our approach offers better F_1 scores for all the 14 gesture classes, except 3. It shows that our model is more balanced than other approaches for DHG hand gesture recognition for most of the gestures. For more detailed results, we present the confusion matrices we obtained with our model on the DHG in table III for DHG14, and in table I for DHG28.

E. Discussion

1) Privacy: In personal and human-related applications, personal information may leak into the model, e.g. due to overfitting issues. Differential privacy techniques such as Differential Stochastic Gradient Descent [1] or Adversarial Training can be used to prevent such leaks. Nevertheless, for applications such as gesture recognition for smart-screens - trained on datasets such as the DHG dataset- slightly overfitting is not an issue from a practical perspective.

2) Dataset Size: A common barrier to using deep-learning is small datasets. The DHG dataset has roughly 3000

	Ours			DE SMEDT *et al.*			Difference
Gesture	Precision	Recall	F_1-score	Precision	Recall	F_1-score	F_1-score
G	72.4%	94.8%	**82.1%**	67.5%	57.0%	61.8%	20.3%
T	71.2%	77.0%	74.0%	85.2%	87.0%	**86.1%**	-12.1%
E	84.7%	90.9%	**87.7%**	84.8%	87.0%	85.9%	1.8%
P	90.9%	78.4%	**84.2%**	52.1%	61.0%	56.2%	28.0%
RC	69.2%	98.2%	**81.2%**	80.0%	77.5%	78.8%	2.5%
RCC	97.8%	77.6%	86.5%	90.9%	85.5%	**88.1%**	-1.6%
SR	91.2%	100.0%	**95.4%**	85.1%	92.5%	88.6%	6.7%
SL	98.0%	88.9%	**93.2%**	78.4%	85.5%	81.8%	11.4%
SU	98.2%	79.4%	**87.8%**	89.3%	85.5%	87.4%	0.4%
SD	93.3%	91.8%	**92.6%**	80.8%	88.0%	84.3%	8.3%
SX	100.0%	89.9%	**94.7%**	95.8%	85.0%	90.1%	4.6%
S+	98.3%	100.0%	**99.1%**	90.2%	98.5%	94.1%	5.0%
SV	90.6%	100.0%	**95.1%**	93.2%	92.0%	92.6%	2.5%
Sh	96.2%	71.4%	82.0%	88.6%	81.0%	**84.7%**	-2.7%

TABLE IV

COMPARISON OF F_1 SCORE IN THE 14 GESTURE CLASSES CASE

F_1 scores by gesture, for a version of the model without the "residual" branch; very similar results are obtained for the model with the "residual" branch.

balanced sequence instances (1960 train sequences + 837 test sequences) of 100 timesteps with 66 1D-channels. The proposed model has 13829454 free parameters in total, or $13829454/66 \approx 209537$ free parameters by channel. Given that each sequence has 100 timesteps, and because of the regularization applied, the model likeky does not overfit. Qualitatively speaking, no overfitting was observed experimentally. The lower bound of the size of datasets needed for deep learning is hard to determine, as it depends on factors difficult to evaluate such as the task complexity, and the model complexity. As a general rule of thumb, without any regularization, one may arguably say that the size of a dataset should be at-least 1 or 2 orders of magnitude than its dimensions.

Zero-, One-, or Few-shot learning [47], as well as data augmentation, transfer learning, model compression and distillation techniques can help to reduce the minimum size of the dataset required for training and validating deep learning models.

3) Preprocessing, Average Pooling & Data Regularity: The input to the network assumes a sequence of poses, which are provided by the Intel RealSense camera. The poses can also be retrieved by using body-worn sensors or estimated by segmenting videos [36].

We re-sampled signals to a vector of size 100 due to the nature of the motions that were all both relatively short as well as all being about the same duration in order of magnitude. This may not hold for motion capture data with very variable time spans for which one may prefer encode with a convolution and memorize with an reccurrent cell like an LSTM or a GRU. Average pooling seems to function better than max pooling on the input data for our model. 1D physical sensors data and 1D motion capture data present more regularity than other 1D data such as text, which means that the data is more compressible (in the time domain). With 1D gesture data it is easier to filter outliers (e.g. because of of physical constraints on the gesture), and outliers have less meaning than outliers in the text domain. For specific

gesture recognition applications involving a lot of semantics like sign languages, such assumptions may probably not hold. We suppose that averaging values from a 1D channel sequence helps to reduce outliers weight with the smoothing. Average pooling may act as a regularizer. As gestures are smooth, averaging the signal probably leads to more signal removal than noise removal. Since we stick to the MC-DCNN framework with an application- and camera-agnostic architecture with no a priori knowledge, our model is directly extensible to other input formats with different 2D or 3D joints. It does not rely much on specific channels, which may occasionally be corrupted in real-world scenarii if the camera does not work perfectly.

4) Recurrence; Speed: One of the goals of this paper was to study if a convolution-only network could lead to state-of-the-art results for gesture classification. While this result is established for short gestures with limited semantic meaning, the question remains open for gestures with very variable time span. For those cases, re-sampling the input might not always work and it might probably more efficient to insert recurrent cells in the model, e.g. after the convolutions, in order to benefit from the (long time range) context by keeping track of the processed input in a memory. In that case, one should carefully check if the model does not overfit, as memory cells are often harder to regularize [42]. Recurrent cells also tends to significantly increase the training time although there is ongoing work, e.g. [4], to alleviate this issue. LSTMs and GRUs can warp time through their gating mechanism, but since CNNs can have gating mechanisms too, it would be interesting to see if gestures with very variable time span and limited semantic meaning could be efficiently classified without involving any recurrent or auto-regressive mechanism. One of the main advanges of using sparse (skeletal) input data instead of dense (image) input data lies in inference speed. On a (good) Intel Xeon CPU E5-1630 v4 @ 3.70GHz processor, without any GPU, the inference time is as low as $\sim 10^{-5}$s for a batch of 32 gestures, which is several orders of magnitude sufficient for real-time applications, even on less efficient processors for embedded systems.

5) Architecture variations; Branch ablation: Very numerous variations on the model architecture are possible, including weights sharing or progressive channels fusion. Regarding the existing model for instance, a partial grid search was the reason behind the choice to use convolution kernels sizes of 3 and 7.

Removing the residual (res. high, low) branch from the model with 14 gestures leads to small a degradition of the accuracy by -1.05% (resp. -0.53%, -1.31%). Though, we can highlight the importance of the three parallel branches: with 28 classes, the model accuracy decreases way more: -5.24% (resp. -6.38%, -4.96%).

V. Conclusion & Future Works

A. Conclusion

We introduced a new convolutional neural network to classify (recognize) hand gestures using skeletal data only. This neural network extends the MC-DCNN framework in several ways. First, it introduces parallel processing branches for each signal. The advantage of two convolutional branches over a single one seems to be that it allows the architecture to access different time resolutions of each signal. Second, the use of residual connection for each signal allows the gradient to better backpropagate in the neural network. Experimentally, it seems to be useful not only regarding the (time of the) training of the network, but also in terms of accuracy results. Finally, dropout is also used as a regularization technique. From a neural network perspective, we observe that (intra- and inter-) parallel processing of sequences using convolutional neural networks can be competitive with neural architectures that use cells specifically designed for sequences such as GRU and LSTM cells. We applied our model to perform hand gesture classification on a challenging hand gesture dataset (DHG dataset). Our method outperforms all existing published methods on this dataset. Our model achieves a 91.28% classification accuracy (+3,04% relative improvement) for the 14 gesture classes case and an 84.35% classification accuracy (+2,45% relative improvement) for the 28 gesture classes case.

B. Future Works

The biggest drawback of our gesture recognition system is that it only works on complete sequences. One way to overcome this issue to get a realtime, step by step, classification could consist in the usage of non-overlapping short time windows. The recognition model would emit a classification of the gesture data inside each window. Finally, the use of an objective function such as the Connectionist Temporal Classification Loss (CTC) [15] could allow an alignment of the classes obtained from the time windows with the desired (actual) ones.

Sharing all the convolutions' weights between all the channels decreases the overall performance of the model, but also greatly decreases the total parameters count. We plan to study the relation between the model accuracy and its total parameters count to find a possibly better trade-off.

Low-level features present more similarities between channels than more abstract, higher-level, features. Sharing the weights of the first convolutions between all channels -for the high-resolution and the low-resolution branches respectively- could probably help to reduce the model parameters' count while keeping an accuracy comparable to the accuracy obtained with the current model.

Another possible follow-up work may involve the introduction of a spatio-temporal attention module over the parallel features extraction module.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[2] P. Bernardis and M. Gentilucci. Speech and gesture share the same communication system. *Neuropsychologia*, 44(2):178–190, 2006.

[3] Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[4] J. Bradbury, S. Merity, C. Xiong, and R. Socher. Quasi-Recurrent Neural Networks. *International Conference on Learning Representations (ICLR 2017)*, 2017.

[5] X. Chen, H. Guo, G. Wang, and L. Zhang. Motion feature augmented recurrent neural network for skeleton-based dynamic hand gesture recognition. *arXiv preprint arXiv:1708.03278*, 2017.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante. A human activity recognition system using skeleton data from rgbd sensors. *Computational intelligence and neuroscience*, 2016:21, 2016.

[8] Q. De Smedt, H. Wannous, and J.-P. Vandeborre. Skeleton-based dynamic hand gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–9, 2016.

[9] Q. De Smedt, H. Wannous, J.-P. Vandeborre, J. Guerry, B. Le Saux, and D. Filliat. Shrec'17 track: 3d hand gesture recognition using a depth and skeletal dataset. In *10th Eurographics Workshop on 3D Object Retrieval*, 2017.

[10] L. Deng, D. Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[11] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, and A. Del Bimbo. 3-d human action recognition by shape analysis of motion trajectories on riemannian manifold. *IEEE transactions on cybernetics*, 45(7):1340–1352, 2015.

[12] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[14] N. A. Goussies, S. Ubalde, and M. Mejail. Transfer learning decision forests for gesture recognition. *The Journal of Machine Learning Research*, 15(1):3667–3690, 2014.

[15] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772, 2014.

[16] I. Guyon, V. Athitsos, P. Jangyodsuk, H. J. Escalante, and B. Hamner. Results and analysis of the chalearn gesture challenge 2012. In *Advances in Depth Image Analysis and Applications*, pages 186–204. Springer, 2013.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] C. Hilverman, S. W. Cook, and M. C. Duff. Hippocampal declarative memory supports gesture production: evidence from amnesia. *Cortex*, 85:25–36, 2016.

[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] A. Klaser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, pages 275–1. British Machine Vision Association, 2008.

[22] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool. Hough transform and 3d surf for robust three dimensional classification. *Computer vision–ECCV 2010*, pages 589–602, 2010.

[23] T. Laurent and J. von Brecht. A recurrent neural network without chaos. 2017.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[26] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2015.

[27] G. Murthy and R. Jadon. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management*, 2(2):405–410, 2009.

[28] N. Neverova. *Deep learning for human motion analysis*. PhD thesis, INSA Lyon, 2016.

[29] N. Neverova, C. Wolf, G. Paci, G. Sommavilla, G. Taylor, and F. Nebout. A multi-scale approach to gesture detection and recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 484–491, 2013.

[30] N. Neverova, C. Wolf, G. Taylor, and F. Nebout. Moddrop: adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1692–1706, 2016.

[31] E. Ohn-Bar and M. Trivedi. Joint angles similarities and hog2 for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 465–470, 2013.

[32] F. J. Ordóñez and D. Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[33] O. Oreifej and Z. Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723, 2013.

[34] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[36] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[38] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *AAAI*, pages 4263–4270, 2017.

[39] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[40] G. Strezoski, D. Stojanovski, I. Dimitrovski, and G. Madjarov. Hand gesture recognition using deep convolutional neural networks.

[41] A. Truong, H. Boujut, and T. Zaharia. Laban descriptors for gesture recognition and emotional analysis. *The visual computer*, 32(1):83–98, 2016.

[42] A. Turkin. Tikhonov regularization for long short-term memory networks. *arXiv preprint arXiv:1708.02979*, 2017.

[43] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.

[45] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 588–595, 2014.

[46] P. Wang, Z. Li, Y. Hou, and W. Li. Action recognition based on joint trajectory maps using convolutional neural networks. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 102–106. ACM, 2016.

[47] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *arXiv preprint arXiv:1707.00600*, 2017.

[48] Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48, 2010.

[49] M. Ye, Q. Zhang, L. Wang, J. Zhu, R. Yang, and J. Gall. A survey on human motion analysis from depth data. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, pages 149–187. Springer, 2013.

[50] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao. *Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks*, pages 298–310. Springer International Publishing, Cham, 2014.