Convolutional Neural Networks for Multivariate Time Series Classification using both Inter- & Intra- Channel Parallel Convolutions

G. Devineau[1] W. Xi[2] F. Moutarde[1] J. Yang[2]

[1] MINES ParisTech, PSL Research University, Center for Robotics, Paris, France
[2] Shanghai Jiao Tong University, School of Electronic Information and Electrical Engineering, China

{guillaume.devineau, wang.xi, fabien.moutarde}@mines-paristech.fr

Abstract

In this paper, we study a convolutional neural network we recently introduced in [9], intended to recognize 3D hand gestures *via* multivariate time series classification.
The Convolutional Neural Network (CNN) we proposed processes sequences of hand-skeletal joints' positions using parallel convolutions. We justify the model's architecture and investigate its performance 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).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.

1 Introduction

Gesture is a natural way for a user to interact with one's environment. 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 on the gesture data, e.g. using a neural network to perform the classification. In this paper we present and study a convolutional neural network architecture relying on intra- and inter- parallel processing of sequences of hand-skeletal joints' positions to classify complete hand gestures. Where most existing deep learning approaches to gesture recognition use RGB-D image sequences to classify gestures [41], 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. Note that the contents of this paper are highly similar to that of [9], especially sections 1, 2 and 3, as well as the figure illustrating the network, however in this article we focus more on practical tips and on justifying the network architecture whereas the original paper focus was more centered on gesture-related aspects. Readers familiar with [9] can directly skip to the subsection *Architecture Tuning* of section IV, in which the network architecture is justified more thoroughly.

2 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 $p_i = (p_i(t))_{t\in\mathbb{N}}$ 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.
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 [19] while several recent surveys on the latter approaches are listed in Neverova's thesis [21].

2.1 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 em-

phasis on geometric features and statistical features. Some commonly used features are the positions 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 [15, 16, 33]. For instance, in [37], Vemulapalli *et al.* propose a human skeletal representation within 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 manifold. Recognition (classification) is finally performed in the corresponding Lie algebra. In [8], 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 [7], 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 into a classifier to perform the gesture recognition. In [5], 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 [40] are Hidden Markov models (HMM), Conditional Random Fields (CRF), discrete distance-based methods, Naive Bayes, and even simple k-Nearest Neighbors (k-NN) with Dynamic Time Warping (DTW) discrepancy.

2.2 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), e.g. using convolutions, for feature extraction and transformation : $f = f_1 \circ f_2 \circ \ldots \circ f_n$.A traditional Convolutional Neural Network (CNN, or ConvNet) model almost always involves a sequence of convolution and pooling layers, that are followed by dense layers. Convolution and pooling layers serve as feature extractors, whereas the dense 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 [18] on RGB-D images. A direct example of hand gesture recognition via image CNNs can be found in the works of Strezoski *et al.* [32] where CNNs are simply applied on the RGB images of sequences to classify. Guerry *et al.* [7] propose a deep-learning approach for hand gesture recognition on the DHG dataset, which is described in section III 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 [29] image classification model is then applied on sequences of 5 concatenated images consecutive in time. In [20], 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 [22, 23]. 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 [38], 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) [12] or Gated Recurrent Units (GRU) [4], 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 [35, 36] 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 [17] 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 [30], 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, we rather seek 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 [42], although this architecture can easily be extended with recurrent cells : [25]. 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 multi-variate sequences such as hand gesture skeleton data. In [9] we introduce a new neural network built upon this framework.

3 Dataset

To evaluate performances of several variations of the proposed neural network model architecture we conducted experiments on the Dynamic Hand Gesture-14/28 (DHG) dataset [7] 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 dataset is split into 1960 train sequences (70% of the dataset) and 840 test sequences (30% test sequences).

4 Parallel Convolutions Model

4.1 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. Our 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.

4.2 Model Architecture

In [9] 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 1. The architecture is inspired by the high-resolution and low-resolution networks from [20]. The use of residual branches in our architecture is inspired from the original Residual Networks paper [11]. 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

1. middle branch in figure 1

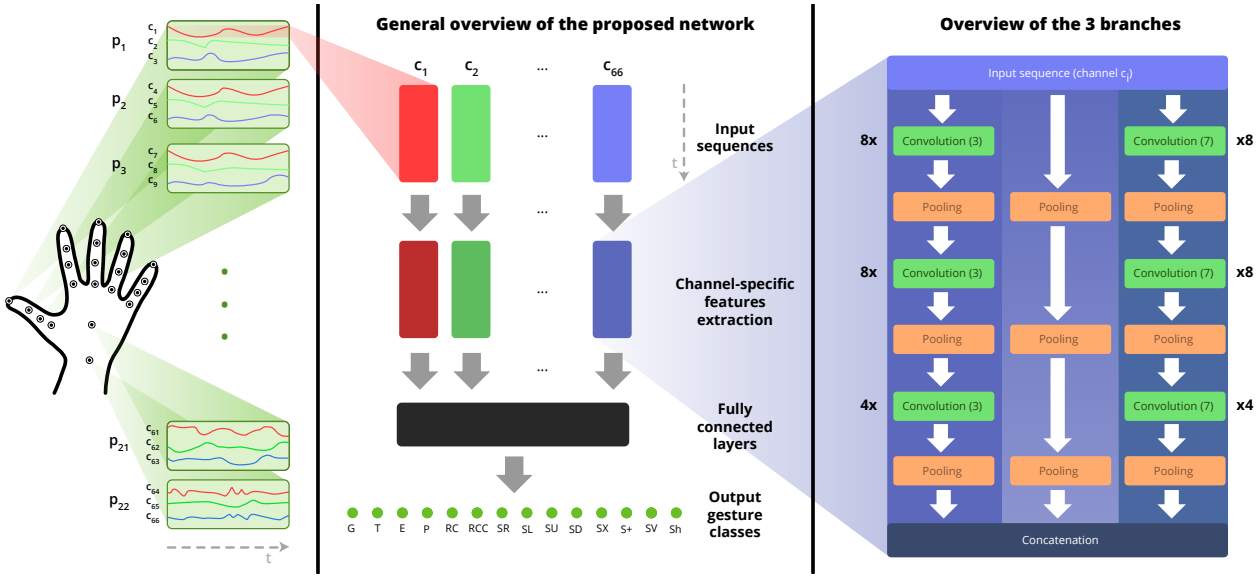

FIGURE 1 – Illustration (borrowed from [9] in which the network architecture is introduced) of the parallel convolutional neural network. Every channel is processed separately before the Multi Layer Perceptron. The parallel feature extraction module presented on the right is not shared between the 66 channels.

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 : $\sigma(x) = \text{ReLU(x)} = \max(0, \text{x})$, 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 [1]. Nevertheless, experiments we conducted on the choice of the pooling method for our model showed that our model exhibits better results with average 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).

4.3 Architecture Tuning

To justify the model architecture, we study the performance of -successive improvements of- a MC-DCNN model, created by tweaking its architecture. We iteratively choose the input data format, the activation function used, the number of convolution layers and feature maps, the regularization rate, the pooling method, and finally we introduce a residual and two parallel CNN branches.

Window length. Our model takes a fixed length gesture sequence. In practice, since the hand gestures from the DHG dataset are relatively short, we do not use a sliding window over the sequences as they occur, but rather simply resample them in time to fit in a fixed time window. To determine the length of the window, we start with a simple mono-branch convolution-pooling model (with 4 convolutions instead of 3 in the final model) and evaluate its validation accuracy depending on the input duration after resampling (see table 1). The window length we chose is 100 timesteps.

Input Length	Train Acc.	Validation Acc.
10	97.30%	82.80%
50	98.88%	84.59%
75	98.42%	86.26%
100	98.78%	**87.10%**
150	98.37%	85.78%
200	98.01%	84.59%

TABLE 1 – Accuracy after training the model for 50 epochs

Activation function. While the choice of the rectified linear unit (ReLU) function is a *de facto* standard in recent deep learning models, each type of activation function has its pros and cons [13]. We thus decided to investigate whether we should use the exponential linear unit (ELU), the rectified linear unit (ReLU) or the sigmoid activation function as the network activation function. As a reminder, these activations are defined as follows : $ReLU(x) = x^+ = \max(0,x)$, $Sigmoid(x) = \frac{1}{1+e^{-x}}$, and $ELU(x) = \begin{cases} \lambda(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$, where $\lambda \geq 0$ is a hyper-parameter to be tuned.
After training the same model for a limited -but sufficient-

number of epochs and evaluating its performances on the validation set (see table 2), we chose to use the ReLU function as the network activation function. ELU are ReLU yield to similar results, although ELU is more computationally expensive. Overall ReLU reveals to be a good choice, one of the reasons being that it introduces sparsity effect on the network.

Activation Function	Training Acc.	Validation Acc.
Sigmoid	7.30%	6.81%
ReLU	91.63%	**83.51%**
ELU	**94.74%**	82.68%

TABLE 2 – Accuracy after training the model for 50 epochs

Other activations worth considering in future works include Leaky ReLUs or Parametric ReLUs, as well as the Swish activation function.
Parametric ReLUs are similar to ReLUs but cells do not "die", which can be the case with ReLU due to the backpropagation issue when $x < 0$. $PReLU(x) = \begin{cases} x & \text{if } x \geq 0 \\ \mu x & \text{if } x < 0 \end{cases}$.
The coefficient μ is learnt during the network training.
The Swish activation function has empirically been shown to display better performances than the ReLU activation function [27] and is defined by $Swish(x) = x \cdot Sigmoid(\beta x)$.

Number of layers. Following a classical convolution-pooling architecture schema, with a single branch and without dropout, we evaluate the performance of the model for a few epochs. The models we evaluate are composed of convolutions with ReLU activations. We denote $C_{ReLU}(8,5)$ (or simply $C(8,5)$) a convolution layer with 8 feature maps and a kernel of width 5, followed by a ReLU activation. We denote $P(2)$ an average pooling layer of size 2. Results are presented in table 3. We chose to use a depth of 3 convolution+pooling layers, since it leads to the best (average) validation accuracy. The depth is likely to be related to the sequence length though, and longer sequences may benefit from being processed by a deeper network, if one wanted to change the input length.

#conv	Architecture	Train acc.	Validation acc.
2	$C(8,5)-P(2)-C(4,5)-P(2)$	94.74%	82.68%
3	$C(8,5)-P(2)C(8,5)-P(2)-C(4,5)-P(2)$	95.71%	**84.95%**
4	$C(8,5)-P(2)-C(8,5)-P(2)-C(4,5)-P(2)-C(4,5)-P(2)$	95.61%	84.59%

TABLE 3 – Accuracy after training the model for 50 epochs

Dropout. To reduce overfitting, we use dropout [31] which consists in randomly setting some layer inputs to 0 at each update during training time only. Results of some experiments we performed to determine whether to apply dropout to the convolutions layers or not, as well as to determine their rate, are presented in the table 4 below (still on a model without parallel branches). We denote $D(0.2)$ the use of dropout in the previous layer with a dropout rate of 20%, which is the rate we finally chose.

Training Acc.	Validation Acc.	Model architecture							
98.98%	87.10%	$C_{ELU}(8,5)$	$-P(2)$	$C_{ELU}(4,5)$	$P(2)$		$C_{ReLU}(4,5)$	$P(2)$	
98.11%	88.41%	$C_{ELU}(8,5)$	$P(2)$	$C_{ELU}(4,5)$	$D(0.2)$	$P(2)$	$C_{ReLU}(4,5)$	$P(2)$	
98.98%	**89.37%**	$C_{ELU}(8,5)$	$P(2)$	$C_{ELU}(4,5)$	$D(0.2)$	$P(2)$	$C_{ReLU}(4,5)$	$D(0.2)$	$P(2)$

TABLE 4 – Acc. after training the models for 100 epochs

Pooling method. Average pooling seems to function better than max pooling on the input data for our model : it outperforms max-pooling by +0.88% in accuracy (result on the 14 classes case). 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.

Residual and Parallel CNN branches. We duplicated the original CNN branch obtained, and add a pseudo-residual branch constituted of pooling only, whose aim is to better backpropagate the gradient during training. To achieve better performance -i.e. better validation accuracy, recall and F_1 scores- we adopt a multi-(temporal-)resolution approach : we chose to change the kernel width depending on the branch (instead of using a kernel of width 5, we use two kernels of sizes 3 and 7).
We highlight the importance of using three parallel branches for the 28 gesture classes case in table 5.

Model	w/o Residual	w/o High Resolution	w/o Low Resolution
14 classes	-1.05%	-0.53%	-1.31%
28 classes	-5.24%	-6.38%	-4.96%

TABLE 5 – Model acc. degradation on branch ablation

4.4 Training

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

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) [10] to set the initial random weights for all the weights of our model.

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

Hardware. We trained our model on one machine with a GPU (NVIDIA GeForce GTX 1080 Ti). Using the hyper-parameters presented in the paper, each training step took about 12 seconds. We trained the model for 1000 steps.

Loss & Optimizer. We selected negative log-likelihood as the cost function. To train our model, we used the Adam optimization algorithm [14] which calculates an exponential moving average of the gradient and the squared gradient. Two parameters, namely β_1 and β_2, are used to respectively weight the first moment (i.e. the mean) and the second moment (i.e. the uncentered variance) of the 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 $\varepsilon = 10^{-8}$. All these values are either identical (α, β_1, ε) or close ($\beta_2 \simeq \beta_2^{(default)} = 0.999$) to the default values proposed in the Adam article.

4.5 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 a 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 (see table 6).

In [26, 8, 7, 24, 6] 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. [3] 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 [3]. 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 6.

4.6 Discussion

Dataset Size, . A common barrier to using deep-learning is small datasets. The DHG dataset has roughly 3000 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 of the 1960 training sequences has 100 timesteps, and because of the regularization applied (DropOut with $p = 0.2$), the model likely does not overfit. Qualitatively speaking, no overfitting was observed experimentally. Zero-, One-, or Few-shot learning [39], 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.

Approaches	Accuracy 14 gestures	Accuracy 28 gestures
Oreifej & Liu [26]	78.53	74.03
Devanne *et al.* [8]	79.61	62.00
Guerry *et al.* [7]	82.90	71.90
Ohn-Bar & Trivedi [24]	83.85	76.53
Chen *et al.* [3]	84.68	80.32
De Smedt *et al.* [6]	88.24	81.90
Ours	**91.28**	**84.35**

TABLE 6 – Accuracy results on the DHG dataset

	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 7 – 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.

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 [28]. In that specific case, occlusion issues may arise, leading to wrong joint positions. Data augmentation may help reduce this problem. Invariance by rotation is not taken into account yet by the network.

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.

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 [34]. Recurrent cells also tends to significantly increase the training time although there is ongoing work, e.g. [2], 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 Conclusion

We introduced and studied a new convolutional neural network which classifies (recognizes) 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 duration of network training, 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% improvement) for the 14 gesture classes case and an 84.35% classification accuracy (+2,45% improvement) for the 28 gesture classes case.

Références

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] G. Devineau, W. Xi, F. Moutarde, and J. Yang. Deep learning for hand gesture recognition on skeletal data. In *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on*. IEEE, 2018.

[10] 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.

[11] 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.

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

[13] G. Jiuxiang, W. Zhen-Hua, J. Kuen, et al. Recent advances in convolutional neural networks, 2015.

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

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. 2018.

[28] 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.

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

[30] 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.

[31] 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.

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

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

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

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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

[41] 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.

[42] 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.