Learning to Recognize Touch Gestures: Recurrent vs. Convolutional Features and Dynamic Sampling

Quentin Debard, Christian Wolf, Stephane Canu, Julien Arné

Learning to recognize touch gestures: recurrent vs. convolutional features and dynamic sampling

Quentin Debard[1,4], Christian Wolf[2,1], Stéphane Canu[3] and Julien Arné[4]
[1] Univ de Lyon, INSA-Lyon, CNRS, LIRIS, F-69621, France
[2] Univ de Lyon, INSA-Lyon, INRIA, CITI, F-69621, France
[3] Normandie Univ, INSA Rouen, UNIROUEN, UNIHAVRE, LITIS, France
[4] Itekube, France

***Abstract*— We propose a fully automatic method for learning gestures on big touch devices in a potentially multi-user context. The goal is to learn general models capable of adapting to different gestures, user styles and hardware variations (e.g. device sizes, sampling frequencies and regularities). Based on deep neural networks, our method features a novel dynamic sampling and temporal normalization component, transforming variable length gestures into fixed length representations while preserving finger/surface contact transitions, that is, the topology of the signal. This sequential representation is then processed with a convolutional model capable, unlike recurrent networks, of learning hierarchical representations with different levels of abstraction.**

To demonstrate the interest of the proposed method, we introduce a new touch gestures dataset with 6591 gestures performed by 27 people, which is, up to our knowledge, the first of its kind: a publicly available multi-touch gesture dataset for interaction.

We also tested our method on a standard dataset of symbolic touch gesture recognition, the MMG dataset, outperforming the state of the art and reporting close to perfect performance.

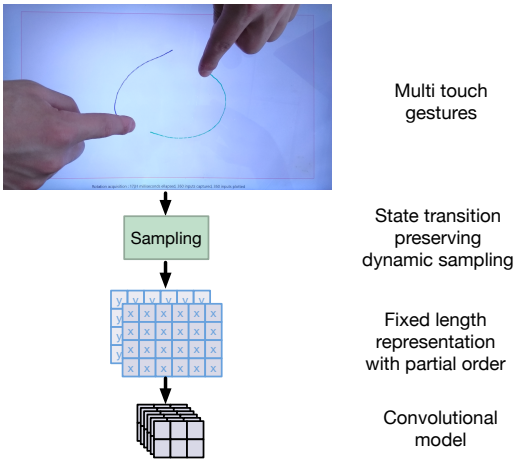

Fig. 1: Overview of the method: multi-touch gestures are dynamically sampled with a transition preserving transform followed by convolutional or 2D-recurrent representation learning.

I. INTRODUCTION

Touch screen technology has been widely integrated into many different devices for about a decade, becoming a major interface with different use cases ranging from smartphones to big touch tables. Starting with simple interactions, such as taps or single touch gestures, we are now using these interfaces to perform more and more complex actions, involving multiple touches and/or multiple users. If simple interactions do not require complicated engineering to perform well, advanced manipulations such as navigating through a 3D modelisation or designing a document in parallel with different users still craves for easier and better interactions.

As of today, different methods and frameworks for touch gesture recognition were developed (see for instance [15], [27] and [8] for reviews). These methods define a specific model for the class, and it is up to the user to execute the correct protocol. Our approach in this paper is to let users define gestures from a simplified protocol. The main motivation is to remove burden from the user and put it onto the system, which shall learn how users perform gestures. The idea is not new and was first explored in 1991 by Rubine [26], using Linear Discriminant Analysis (LDA) on 13 handcrafted features. Although other methods discussed in section II have built on this idea, our goal is to generalize it even further by learning deep hierarchical representations automatically from training data.

To achieve these goals, our method must capture the particularities of each gesture class while at the same time being robust with respect to variations in sampling frequencies, screen size, user preferences. A further constraint is the computational complexity of the prediction model, since decoding our model must be possible in real-time applications.

To address these issues, we propose a new method and provide several contributions:

- We address the problem of classifying sequential data characterized by variable amounts of data per time instant. We propose a convolutional model for this problem, and show that it is able to outperform classically used recurrent neural networks on this task. In contrast to recurrent models, our convolutional model is able to create hierarchical representations encoding different levels of abstraction.
- We propose a novel input sampling method which drastically reduces the length of the input sequences while at the same time preserving important sharp transitions in the data.

- We propose a new dataset of multi-touch sequential gestures, which is, up to our knowledge, the first of its kind. Existing datasets are restricted to symbolic gestures. The dataset will be made publicly available on acceptance of the paper.
- We also compare the method against the state of the art on a standard dataset in touch gesture recognition.

II. Related work

Automatic human gesture recognition is an extremly prolific field. Using many different sensors such as RGB cameras, depth sensors, body sensors, in our case touch surfaces, these gestures are classified and measured through representations: geometric features, graphs, state machines, sequences, and more recently, learned features. The classifying and measuring algorithms are also varied, ranging from Deterministic Decision Models to Support Vector Machines and Deep Neural Networks.

Touch gestures — can be distinguished into two types:

- **symbols**, such as drawings or handwriting. These gestures are spatially complex, but their temporal properties are of lesser interest.
- **interactions**, meant to perform an action using the touch surface as an interface. These actions require spatial and temporal precision, as the user will expect the interaction to be as precise and fast as possible.

Touch gestures are traditionally dealt with handcrafted representations. The most commonly used methods have been developed by system designers, using procedural event-handling algorithms (see for instance [30] or [22]). Different frameworks such as Gesture Markup Language (GestureML) were proposed in order to formalize touch gesture interactions. Midas [27] uses a set of logical rules to classify events on the surface. With the possibility to define custom operators and priority rules, its gesture definition is extensible to some point, but lacks rotation invariance. Proton++ [15] is another framework, based on regular expressions for gesture recognition: a gesture is seen as a sequence of events. However, it only supports a unique gesture at a time, and is limited by the rigidity of regular expressions.

As efficient and fast as they can be, these methods arbitrarily declare gesture properties, making the user adapt to them. The gestures are precisely defined and tend to lack generalization in a different context; this contradicts our paradigm of minimal user constraint and maximum generalization.

In contrast to these user-defined gesture frameworks, Rubine in 1991 developed a more flexible gesture definition [26], using handcrafted geometric features and LDA for classification. Up to our knowledge, this is the first attempt at classifying gestures using deep learning. Gesture Coder [21] takes a similar approach as Proton++, as it defines gestures using state machines, equivalent to regular expressions on "atomic actions". However, these state machines are learnt from user gestures. [6] uses a graph representation of gestures, then embeds the graph structure into a feature vector. These feature vectors are then classified using a Support Vector Machine. We also recommend [8] as a good survey of the evolution in multi-touch recognition.

Visual / Air gestures — are gestures performed without any touch surface and captured by video cameras or depth sensors. We mention these methods here, since a part of the methods described in the literature can be adapted to touch gesture recognition. We will only briefly mention methods using handcrafted representations, which normalize an articulated pose estimation (skeleton) into a view- and person-invariant description, followed by machine learning [32], as well as methods based on deep neural networks working on pose fused with raw video [24].

Sequential models — are the main methodology for gesture recognition, gesture data being of sequential nature. One of the first successful statistical models used are Hidden Markov Models (HMMs) [25], which are generative probabilistic graphical models with a linear chain structure. The hidden state of these models is stochastic, therefore, in the most frequent variants, training and decoding requires to solve combinatorial problems. Conditional Random Fields (CRFs) [18] are discriminative counterparts of HMMs. The modeled distribution is conditioned on the observations, which allows the model to concentrate its capacity on the discrimination problem itself.

Recurrent Neural Networks (RNNs) are the connectionist variant of sequential models introduced in the 80s. Their hidden state is rich and componential and not stochastic, which allows to decode the model in a forward pass as a computation in a direct acyclic graph. An important variant of RNNs are Long Short-Term Mermory networks (LSTMs) [12], which models additional long term transitions through gates in the model, and their simplier variations GRU [7]. Clock-work RNNs, introduced in [16], introduce sampling and update frequencies into recurrent networks, allowing for a hierarchical decomposition of the signal. Dense CWRNN adapt this representation to shift invariant models [23].

III. Recurrent vs. convolutional models

Our problem can be cast as a sequential learning problem with input dimensions varying over time. Each input gesture is a sequence $(\boldsymbol{u}^n_{t,i})_{t=1,T_n}$ of length T_n where n is the index of the touch gesture, i the finger ID provided by the touch device and $\boldsymbol{u}^n_{t,i} = (x^n_{t,i}, y^n_{t,i})^T$ are the spatial 2D coordinates of finger i on the touch screen. An example of such a sequence is illustrated schematically in Figure 2a. Note that a variable amount of fingers may touch the surface. Therefore, finger ID indexes $i \in \{1 \dots I_n\}$, I_n being the number of fingers involved in gesture n. Finger IDs are from an unordered set and provided directly by the touch hardware. We suppose that finger tracking allows finger IDs to be identical for the same finger as long as the finger touches the screen; however, removing a finger and putting it on the screen again will not generally preserve its ID. A similar approach was taken in [26]. In the following, gesture indices n can be omitted for clarity, unless necessary for comprehension.

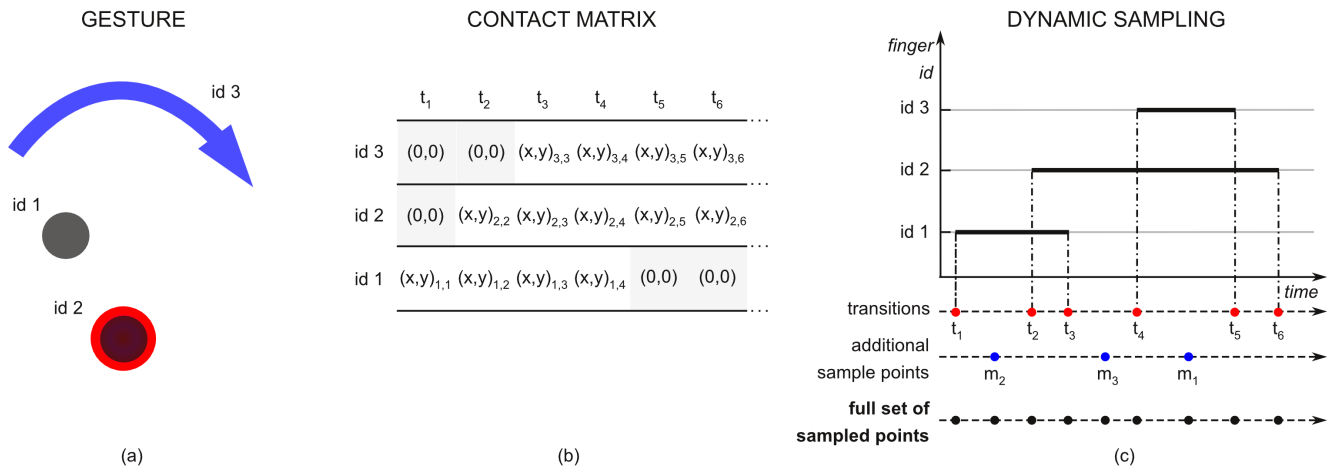

Fig. 2: From raw data to gesture representations: (a) a graphical representation of a gesture, where red presses are held throughout the gesture; grey touches are taps: quick, almost immobile contacts with the surface; blue touches are slides. (b) the device delivers sequences of 2D coordinates (x, y) over time and for tracked finger IDs; (c) Our sampling procedure (section IV) selects samples time @instants t at finger state transitions (in red) and additional sample points (blue).

We address the problem of joint learning of a prediction model for classification together with a feature representation from training data. The main difficulty we face is the fact that the data is temporal with a variable number of data points (fingers) at each time step, which makes it difficult to align off-the-shelf sequential models like RNNs and their variants directly on the input feature dimension of the data.

One strategy is to train a model which integrates data points $\boldsymbol{u}_{t,i}$ for a single time instant t into a fixed length representation, through a learned mapping $f_t = \phi(\boldsymbol{u}_{t,1:I_t}, \theta_\phi)$ parameterized by θ_ϕ (gesture index n has been omitted). Again, the index $1{:}I_t$ indicates the number of inputs, which is varying depending on t. This makes it difficult to learn this mapping with classical models, which suppose that the data are embedded in a vector space. In our case, each data point is of fixed length, but the set indexed by $1{:}I_t$ is not.

Handcrafted representations could be designed, to embed this set of samples into a fixed length representation which describes the spatial distribution of the points. In the literature, several representations have been proposed, but we will only mention Shape Context [4]. In [4], log polar histograms are computed for a point cloud, which describe the positions of individual points relatively to other points in the cloud. In our work, we prefer to automatically learn a suitable feature representation from data using a sequential model.

It is very important to remark here that a sequential model is trained on data which is *unordered*. In other words, the model is required to ignore the evolution of the data over its finger ID dimension, since the input data is not ordered in this dimension. The model will need to learn to embed the data into a spatial representation, in a similar spirit as Shape Context histograms. Ensuring invariance to finger order is therefore important, whose learning can be favored with data augmentation, i.e. shuffling finger IDs during training.

The resulting features f_t can then be integrated temporally using a second sequential model $a{=}\psi(f_{1:T}, \theta_\psi)$, parametrized by θ_ψ, which operates in the time dimension and predicts a gesture class a for the sequence. This model is illustrated in figure 3a, which shows the two mappings $\phi(.)$ and $\psi(.)$ as recurrent networks with respective hidden units h and h'. Here, features f_t are outputs of $\phi(.)$ — an alternative choice would have been to use the hidden representation h itself as features input to $\psi(.)$.

The model described above corresponds to the basic requirements of a model responding to the given problem. In the rest of this section, we will provide two deep neural models extending this principle, namely recurrent networks and convolutional neural networks. We will argue the superiority of convolutional features, and in section VI we will confirm these arguments through experiments.

A multi-dimensional recurrent model — As mentioned above, touch finger IDs are not ordered; however, fingers are usually tracked over time by the hardware. Finger IDs are therefore consistent over time, at least between finger state transitions. The model described above does not directly describe transitions of individual finger inputs over time. Of course the representation mapping $\phi(.)$ can theoretically learn a representation f_t, which allows the temporal mapping $\psi(.)$ to disentangle individual fingers and track their temporal evolution. In practice, learning this is unnecessarily hard.

A different strategy is to create trainable and direct temporal transitions between successive samples of the same fingers. This requires to handle transitions in two different dimensions: finger ID and time. Using connectionist learning frameworks, one possible strategy is to use multi-dimensional RNNs or LSTMs [10][20], a straight-forward extension of the 1D original models, or closely related variants like Grid RNNs [14]. In these models, the hidden state (and also the hidden memory cell in the case of the LSTM variant of the model) of time t is not only computed from one single predecessor $(t{-}1)$ but from several predecessors, one for each dimension. This is illustrated in figure 3b for

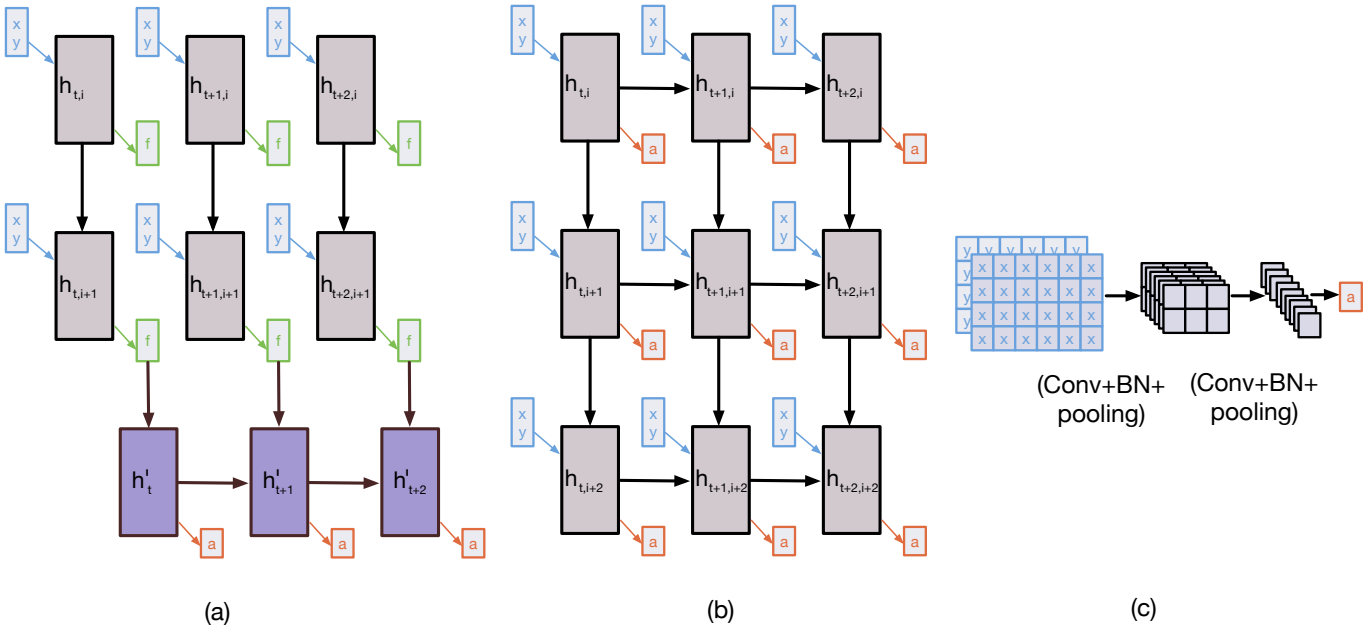

Fig. 3: Feature representations for sequences: (a) sequential learning of fixed-length representations; (b) MD-RNNs / MD-LSTMs (gates are not shown); (c) convolutional features for sequential data. Input data is shown in blue, output in red (best viewed in color).

two dimensions (finger ID and time).

This model can handle data of variable length in time and of variable numbers of fingers per time instant. However, when finger state transitions (finger press/release events) occur, partially empty data rows are created, which need to be padded, for instance with zero values.

A convolutional model — gestures are characterized through their short term behavior as well as their (relatively) long term behavior. A good model should be able to capture the former as well as the latter, i.e. the statistical long range dependencies in the input data. In principle, the recurrent models described above are able to do that. However, since they satisfy the Markov property (the state of the model at time t depends only on the state of the model at time $t-1$ and not on the preceding states), all long range dependencies need to be captured by the hidden state.

Convolutional Neural Networks (CNNs), on the other hand, have proven to be efficient in learning hierarchical hidden representations with increasing levels of abstraction in their subsequent layers. Traditionally, these models have been applied to images [19], where filters correspond to spatially meaningful operations.

Very recently only, they have been reported to be efficient for learning hierarchical features on sequential data, for instance for human activity recognition from articulated pose (skeletons) [3] or for machine translation [9]. In these cases, the temporal dependencies of the data are not covered by a unique hidden state, but by a series of feature maps, which capture temporal correlations of the data at an increasing level of abstraction. While the first filters capture short term behavior, the last layers capture long range dependencies.

In this work, we propose a similar strategy for the recognition of touch gestures. The proposed CNN uses 2D spatial filters. The sequential input data is structured into a 3D tensor, with the finger ID as first dimension, time as the second dimension, and input channels as the third dimension (x coordinates are in the first channel and y coordinates in the second channel) — as illustrated in figure 3c. The 2D convolutions therefore capture correlations in time as well as correlations over finger IDs.

Data preparation — in this work we propose convolutional features for touch gesture recognition, and in the experimental section we will compare them to MD recurrent models described above, as well as to the state of the art in touch gesture recognition.

CNNs require fixed length representations[1]. We therefore propose a novel feature preserving dynamic sampling procedure, which will be described in the next section.

IV. Transition preserving dynamic sampling

The input gestures are of variable length, of varying numbers of fingers per instant, and eventually of varying speed, which is due to variations in user behavior but also to differences in sampling rates. In practice, we observed samples rates between 5 ms and 25 ms. Varying speed can be dealt with easily and traditionally by delegating it to the model and to the training procedure using data augmentation, and/or by using temporal multi-resolution representations. Varying numbers of fingers is coped with by padding. We define a maximum number of fingers depending on the application, as shown in Figure 2, and zero pad the unused entries.

[1]Fully convolutional architectures withstanding, which are predominant in segmentation applications.

Algorithm 1: Dynamic sampling of K-points

Input: a gesture $U = \{\boldsymbol{u}_{t,i}\}, t \in [0, T]$
transition indicators $O = \{\boldsymbol{o}_t\}, t \in [0, T]$
Output: Ordered and indexed set of sample points C

1) $C = \{0\} \cup \{t : \boldsymbol{o}_t = 1\}$

2) **While** $|U'| < K$:
$k = \underset{x \in [0, T-1]}{\arg\max} |C_{x+1} - C_x|$
$C = C \cup \lfloor \frac{k}{2} \rfloor$

Variable temporal length is a different issue, and is a hard problem for convolutional models. In theory it can be dealt with sequential models, like RNNs and their variants. In practice, effects like vanishing/exploding gradients [5] tend to limit these models to relatively short sequences, despite efforts to tackle these limitations [12][7].

Existing work tends to perform sampling spatially, as for instance in [29]: because the task in these papers is to classify symbolic gestures, temporal features such as state transitions or velocity are of minimal interest. We call state transition the moment when at least one finger is added onto or withdrawn from the touch surface. When the task involves interaction gestures [6] [21], dimension reduction is often done through feature extraction/embedding, not sampling.

We chose to normalize the data through a feature preserving transform which compresses it into a fixed length representation. State transitions, which are key features in touch gesture recognition [21], are preserved with this sampling strategy. For gestures with high temporal content, where spatiality does not alter much the classification (such as a press tap, see Figure 4), missing one of these transitions will most likely result in a misclassification of the gesture. Using a uniform sampling, quick transitions such as a tap can be missed.

The goal is to transform a variable length sequence $(\boldsymbol{u}^n_{t,i}), t = \{1..T^n\}$ into a fixed length representation $(\boldsymbol{u}'^n_{t,i}), t = \{1..K\}$. We perform this by choosing N sampling time instants t which are common over the finger IDs i. The set sampling points should satisfy two properties:

i) the points should be spaced as uniformly as possible over the time period;

ii) the sampled signal should preserve finger transitions, i.e. transitions (finger up or finger down) should not be lost by the transform.

To formalize this, we introduce a transition indicator variable o_t defined as follows: o_t=1 if a transition occurs at time t (finger touch down or finger release), and o_t=0 else. Then, the *inverse* problem, namely creating observed gesture sequences from a set of given sample points, can be modeled as a probabilistic generative model, in particular a Hidden Semi-Markov Model [31] (HSMM) with explicit state duration modelling. Obtaining samples from the observed sequence then corresponds to decoding the sequence of optimal states.

In this formulation, the indicator variables o_t correspond to the observations of the model, whereas the hidden state variables S_t correspond to the sampling periods. Each state variable can take values in the set of hidden states Λ=$\{1..K\}$, which correspond to the K target samples. The desired target sampling points correspond to instants t where changes occur in the hidden state S_t. The transition function of the model is a classical forward model (upper case letters indicate random variables, lower case letters realizations):

$$\begin{aligned} \boldsymbol{q}_{(i,d,j)} &\triangleq P(S_{[t+1:[}=j|S_{[t-d+1:t]} = i) = \\ &= \begin{cases} \frac{1}{S} & \text{if } j{=}i+1 \\ 0 & \text{else} \end{cases} \end{aligned} \tag{1}$$

The duration probability distribution encodes above property (i), which aims sampling points equally spaced with a target period of $\frac{T^n}{K}$:

$$\begin{aligned} \boldsymbol{p}_{j,d} &\triangleq P(S_{[t+1:t+d]}{=}j|S_{[t+1:[} = j) = \\ &= \tfrac{1}{Z}(d - \tfrac{T^n}{K})^2 \end{aligned} \tag{2}$$

where Z is a normalization constant.

The observation probabilities encode the hard constraints on the transitions (above property (ii)):

$$\begin{aligned} \boldsymbol{b}_{j,d}(o_{t+1:r+d}) &\triangleq P(o_{[t+1:t+d]}|S_{[t+1:t+d]} = j) = \\ &= \begin{cases} 0 & \text{if } \sum_{j=t+1}^{r+d} o_j > 1 \\ \frac{1}{Z'} & \text{else} \end{cases} \end{aligned} \tag{3}$$

where Z' is a normalization constant. In other words, sampling periods spanning over more than 1 transition are forbidden, which makes it impossible to lose state features.

If the number of transitions is lower than the number K of desired sampling points, then the Semi-Markov model parametrized by the above equations[2] can be decoded optimally using the Viterbi algorithm [31]. Because the complexity is high, we solve the problem faster and heuristically with a greedy algorithm, which first selects all transition points for sampling, and then iteratively adds additional sampling points in the longest remaining sampling intervals (see algorithm 1).

A drawback of the proposed sampling method is the variations in the sampling rate over gestures: since the sampling is not uniform, we lose velocity information if we only consider spatial coordinates. One possibility would be to keep time stamp information additionally to coordinates, making it possible for the model to extract the required local velocity. In practice, experiments showed that the resulting gestures are sufficiently equally sampled and adding time stamps did not improve performance.

V. The Itekube-7 Touch Gesture Dataset

We introduce a new touch gesture dataset containing 6591 gestures of 7 different interaction classes (illustrated in

[2]For space reasons, we omitted the initial conditions which ensure that the optimal sequence begins with state 1 and terminates with state T.

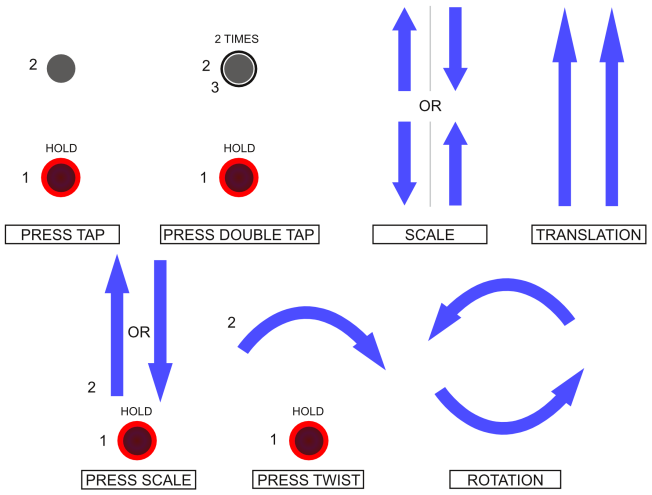

Fig. 4: The classes of the proposed novel multi-touch gestures dataset. Red presses are held throughout the gesture. Grey touches are taps: quick, almost immobile contacts with the surface. Blue touches are slides.

Figure 4) performed by a total of 27 different people. These persons are from different professional backgrounds and aged from 12 to 62. The dataset is available at `http://itekube7.itekube.com`.

Samples correspond to finger contacts on the touch surface. A sample contains the finger ID (provided by the hardware), x and y coordinates and a timestamp. A finger is tracked as long as it stays in contact with the surface. Coordinates are normalized with respect to the screen size, from 0 (top-left) to 1(bottom-right).

The gestures descriptions provided to the users were deliberately minimal, in order to grasp as many user variations as possible. The gestures can be executed anywhere on the screen, with any orientation, scale or velocity. Users were asked to perform the gesture naturally, we did not insist on a very strict definition of the finger state transitions. It means the user knows the different classes, and performs each one as he wants as long as we can distinguish the different classes. In consequence, some classes can be defined by different transition sequences; for example, on press tap, some users lift the press finger first, whereas others lift the tap one first. Some classes are highly correlated: press tap and press double tap are only distinguishable from their transitions, press scale and scale differs from one trajectory. From our experiments, these two classes were usually the hardest to separate (see Table II).

VI. Experimental Results

We tested our method on 2 different datasets: the Itekube-7 Touch Gesture Dataset introduced in section V and the Mixed Multistroke Gestures (MMG) dataset [2]. The latter dataset contains 9600 gestures from 20 participants in 16 classes. All classes are symbol gestures: the sequence order has no importance for the classification, we classify the symbol drawn by all the trajectories. Each class is performed 10 times by a participant at three different speeds, resulting in 30 occurrences per class per participant.

Experimental setup — Because of hardware variations, we want our model to be invariant to finger IDs: we should avoid any correlation between a gesture and the finger ID provided by the device, which results in a finger order in the input tensor. To address this problem, we perform data augmentation on the ID permutations: each gesture of the training set is augmented by permutating its lines (corresponding to a single finger trajectory). For our multi-touch dataset, we set the maximum number of fingers to 3 and thus keep all 6 permutations of each gesture.

The coordinates (x, y) of each datapoint have been normalized on their respective axis between 0 and 1, for any device. This means that depending on the screen size, gestures will be relatively larger, smaller or even distorted if the ratio is different. Normalization proved to be inferior to data augmentation on this problem; we therefore artificially increase the scale variation of the dataset. Each gesture permutation is rescaled two times between 0.5 and 1.5. This brings the size of the augmented training set to 12 times the original one. Because of the freedom given to the subjects to perform gestures, data augmentation on gesture orientation was not necessary.

Architectures and implementation details — We implemented our model in Tensorflow [1]. All hyper-parameters have been optimized over the validation set, the test set has not been used for this. The number of sampled points set to $K{=}10$ was the best trade-off between information maximization and redundancy minimization for our problem.

- For readability purposes, we refer to convolutional layers with x feature maps as CONVx layers, and max pooling layers as POOL. The convolutional model has the following architecture: a CONV128 layer, a $1{\times}2$ POOL layer (max pooling only on the time dimension), again a CONV128 layer and a $1{\times}2$ POOL layer, a CONV256 layer and a fully connected layer providing the prediction score for each class. Activation functions are ReLU[17], all convolutional kernels are $3{\times}3$. The fully connected layer is linear (no activation function). Dropout is set to 0.5. The network is further normalized using batch normalization [13]. The model has been trained for about 400 epochs using a learning rate of 0.001 and an Adam optimizer with decay rates of 0.9 (beta1) and 0.999 (beta2).
- The LSTM used in Table I is the standard version of [12], trained for 300 epochs. For this model all 3 (x, y) coordinate pairs are concatenated to produce a 6 dimensional feature vector. There are 128 hidden units for a cell, and a fully connected layer is used to linearly activate the output.
- For the 2D Spatio-Temporal LSTM we used the variation of [20], which is itself a variant of the Multi-dimensional LSTM[10]. [20] uses a "trust-gate", which filters the input in order to compensate for noise.
  We apply recurrent dropout as defined in [28]. It is trained for 150 epochs. In our model, each cell pos-

	Methods	Sampling	Data augmentation	Accuracy
A	LSTM [12]	-	X	58.71
B	LSTM [12]	Dynamic	X	73.10
C	2D-LSTM [20]	-	X	60.01
D	2D-LSTM [20]	Dynamic	X	87.72
E	Convolutional model	-	-	65.96
F	Convolutional model	-	X	73.00
G	Convolutional model	Uniform	X	80.95
H	Convolutional model	Rand. Uniform	X	80.62
I	Convolutional model	Dynamic	-	83.93
J	Convolutional model	Dynamic	X	**89.96**

TABLE I: Results on the proposed multi-touch dataset: different sequential models and ablation study.

sesses 64 hidden units and the trust parameter is set to 0.5. An activation layer takes all cell outputs (from the whole grid) to compute predictions.

For every model, we use mini-batches of 64 gestures. We use softmax on the output layers, and cross-entropy loss.

Evaluation protocol — We report classification accuracy on the test set, which has been used neither for training nor for architecture and hyper-parameter optimization. The split between test data, validation data and training is subject wise. No subject (person) is in more than one subset of the data.

All optimizations have been optimized using validation error, which is measured with the leave-one-subject-out (LOSOCV) protocol common in gesture recognition (test data is *not* used in this protocol). After optimization of architectures and hyper-parameters, the full combined training+validation set was used again for retraining the final model tested on the test set.

Ablation study — In order to assess the effectivness of each part of the process, we proceed to an incremental evaluation of our method. All the results are displayed in Table I.

- The two recurrent baselines (LSTM and 2D-LSTM [20]) perform worse than the convolutional model, which confirms the reasoning that hierarchical respresentations over time are useful for sequences. However, 2D-LSTMs do have several interesting properties, while performing close to the convolutional model (-2 points). They use only 138,631 trainable weights in total (against 446,983 of the CNN), and they can be unrolled on varying input dimensions. In general, recurrent models are more easily generalizable while CNNs tend to perform better for this task.
- We trained a model without transition preserving dynamic sampling. To this end, and in order to still have a fixed length representation for the convolutional model, the sequences were cropped or padded to 104 points, which is equivalent to 1200ms. This value is fitting 95% of all the dataset gestures. We observed that longer gestures were most likely held for too long or noisy. The architecture was optimized for this experiment (again on the validation set), which resulted in two additional convolutional layers with pooling which are able to cope with these longer sequences. We added 1×3 max-pooling over the time dimension. With 65.96% on the test set, the model performs much worse than the version with dynamic sampling. With data augmentation, the recognition rate rises to 73.00%, but is still far from the 89.96% we obtain with dynamic sampling.
- Data augmentation is an important part of the method, which increases the invariance of the respresentation with respect to the order of the finger IDs delivered by the device, as well as the scale of the gestures. The best performing model w/o augmentation scores at 83.93%, almost 6 points below the best performance with augmentation. Our sampling method was also compared to a uniform and a randomized uniform sampling. This last sampling method was defined by uniformly segmenting the sequence, and then picking a sample from each segment using a normal distribution.

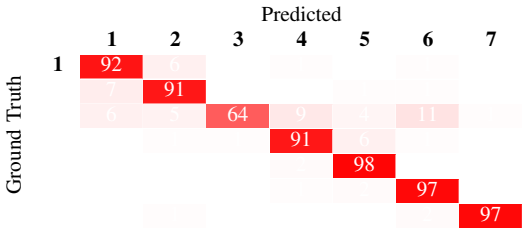

TABLE II: Confusion matrix on the test set for our dataset. Classes are: Press Tap, Press Double Tap, Press Scale, Press Twist, Rotation, Scale, Translation

Methods	Evaluation protocol Leave-one-out	User-independent [2]
Proposed method	98.62	99.38
Greedy-5 [2]	N/A	98.0

TABLE III: Results on the MMG Dataset [2].

Compared to a simple LSTM, A stacked LSTM of 472,839 weights only gains 0.55 points, while a CNN of 112,903 parameters loses only 2.97 points compared to our standard CNN. This confirms that the result gap between the two models is not correlated to the number of weights.

Comparison with the state of the art — We applied the method on the MMG dataset, one of the very few standard datasets of this problem. On this dataset, gestures are complex drawings with a finger or a stylus, performed with a varying number of strokes for a same class. This problem can be seen as symbol recognition. As such, state transitions are not relevant for this task, because temporality does not give meaningful information. We therefore detected geometric transitions as spatial discontinuities (corners) in the finger trajectories. To this end, we calculated thresholded angles of the spatial gradient and thresholded derivates of these angles. We then sampled 48 points of each gesture using the feature preserving method given in section IV (as opposed to uniform sampling of 96 points done in [2]).

For this dataset, we chose an architecture similar to the 6-layers CNN for our own dataset. However, convolutions are 1D, as there is only one stroke at one time on the surface. The architecture is described as follows: 2x(CONV-128+POOL), 2x(CONV-256+POOL), CONV-512, 1x FC. The first CONV layer uses a kernel of size 5 while the others use a kernel of size3. There are a total of 747,536 trainable parameters.

Table III presents the results on this dataset. We used different evaluation protocols: there is no test set for this dataset, so we used the LOSOCV protocol described above for our validation error, as well as the user-independent protocol used in [2]. In this protocol, a user among 20 is randomly selected as the test subject, while the training is performed on the 19 other users. The training is done using 9 samples from every class randomly taken from every training user. We then classify one random sample of every class from the test subject. This process is performed 100 times and classification results are averaged. The obtained performance of 99.38% is close to perfect recognition and beats the state of the art of 98.0% given in [2]. This further confirms the interest of our model and sampling procedure.

Runtime complexity — All computations were done on Nvidia Titan-X Pascal GPUs. Training the convolutional models on our dataset takes 1h11min ($\sim$400 epochs). Testing a single gesture takes 1.5 ms, including the sampling procedure. To assert the portability of this model, runtime on CPU (Intel i7-7700HQ) is 5ms, only 3.3 times slower than on GPU. This is because input tensors are small, resulting in limited GPU acceleration, I/O being the bottleneck.

VII. Conclusion

We have proposed a novel method for multi-touch gesture recognition based on learning hierarchical representations from fixed length input. We have also introduced a dynamic sampling algorithm which preservers sharp features in the input data. We validated the method on our dataset of interaction gestures and on an existing symbolic gesture dataset. This work is the first step toward a rich and adaptative model for touch surface interactions. The runtime complexity of the method allows for the development of real-time applications.

The next challenge is the segmentation of a data stream in order to recognize multiple gestures at once. This will open our research to multi-user interactions and long dependency gestures. In order to process gestures that require an interface update while they are being performed (holding and moving an object in an environment for example), we will focus at some point on early detection processes [11]. Another perspective is reinforcement learning in order to deal with complex decisions while adapting to model its environment and to user styles.

References

[1] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] L. Anthony and J. Wobbrock. $n-protractor: a fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012*, GI 2012, pages 117–120, 2012.

[3] F. Baradel, C. Wolf, and J. Mille. Pose-conditioned spatio-temporal attention for human action recognition. *arxiv:1703.10106*, 2017.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *TPAMI*, 24(24):509–521, 2002.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, 1994.

[6] Z. Chen, E. Anquetil, H. Mouchere, and C. Viard-Gaudin. A Graph Modeling Strategy for Multi-touch Gesture Recognition. *International Conference on Frontiers in Handwriting Recognition*, 2014-December:259–264, 2014.

[7] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[8] M. Cirelli and R. Nakamura. A survey on multi-touch gesture recognition and multi-touch frameworks. ITS '14, pages 35–44, 2014.

[9] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122, 2017.

[10] A. Graves, S. Fernández, and J. Schmidhuber. Multi-dimensional recurrent neural networks. *ICANN*, (1):549—-558, 2007.

[11] M. Hoai and F. De la Torre. Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202, 2014.

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

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[14] N. Kalchbrenner, I. Danihelka, and A. Graves. Grid Long Short-Term Memory. In *ICLR*, page 14, 2016.

[15] K. Kin, B. Hartmann, T. DeRose, and M. Agrawala. Proton++: A Customizable Declarative Multitouch Framework. *UIST*, pages 477–486, 2012.

[16] J. Koutník, K. Greff, F. J. Gomez, and J. Schmidhuber. A clockwork RNN. *CoRR*, abs/1402.3511, 2014.

[17] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105. 2012.

[18] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling data. In *ICML*, 2001.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[20] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. 2016.

[21] H. Lü and Y. Li. Gesture Coder: A tool for programming multi-touch gestures by demonstration. *SIGCHI Conference on Human Factors in Computing Systems*, pages 2875–2884, 2012.

[22] S. Malik, A. Ranjan, and R. Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *UIST*, pages 43–52, 2005.

[23] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, G. Taylor, and F. Nebout. Learning human identity from motion patterns. *IEEE Access*, 4:1810–1820, 2016.

[24] N. Neverova, C. Wolf, G. Taylor, and F. Nebout. Moddrop: adaptive multi-modal gesture recognition. *TPAMI*, 38(8):1692–1706, 2016.

[25] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[26] D. Rubine. Specifying gestures by example. In *Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 329–337, 1991.

[27] C. Scholliers, L. Hoste, B. Signer, and W. De Meuter. Midas: A declarative multi-touch interaction framework. In *International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '11, pages 49–56, 2011.

[28] S. Semeniuta, A. Severyn, and E. Barth. Recurrent dropout without memory loss. *CoRR*, abs/1603.05118, 2016.

[29] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a 1 recognizer for user interface prototypes. *UIST*, 85(2):159, 2007.

[30] M. Wu and R. Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *UIST*, pages 193–202, 2003.

[31] S.-Z. Yu. Hidden semi-markov models. *Artificial Intelligence*, 174:215–243, 2010.

[32] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The Moving Pose: An Efficient 3D Kinematics Descriptor for Low-Latency Action Recognition and Detection. In *ICCV*, 2013.