Procedia Computer Science

Volume 53, 2015, Pages 383–390

2015 INNS Conference on Big Data

A New Multi-Channels Sequence Recognition Framework using Deep Convolutional Neural Network

Runfeng Zhang[1], Chunping Li[2], and Daoyuan Jia[3]

[1] School of Software, Tsinghua University, Beijing BJ 100084, China
rf-zhang12@mails.tsinghua.edu.cn
[2] School of Software, Tsinghua University, Beijing BJ 100084, China
cli@tsinghua.edu.cn
[3] School of Software, Tsinghua University, Beijing BJ 100084, China
jiady14@mails.tsinghua.edu.cn

Abstract
Nowadays, a variety of sequences could be recorded and used with the rapid development of intelligent devices and sensors' integrated technology. Several analysis of the sequences are based on the sequence recognition or classification and most of them are implemented via traditional machine learning models or their variants, such as Dynamic Time Warping, Hidden Markov Model and Support Vector Machine. Some of them could achieve a relatively high classification accuracy but with a time-consuming training process. Some other models are just the opposite. In this paper, we proposed a novel framework to solve the recognition task for sequences with multi-channels with a higher accuracy in less training time. In our framework, we designed a novel deep Convolutional Neural Network using "Data-Bands" as inputs. We conducted contrast experiments between our framework and several baseline methods and the results demonstrate that our framework could outperform state-of-art models.

Keywords: Deep Learning, Sequence Recognition, Machine Learning

1 Introduction

Nowadays, intelligent devices and modern life are more and more inseparable. These devices could record a variety kinds of sequences through embedded sensors precisely. Lots of sequence recognition models have been proposed, and most of them made use of Hidden Markov Models, Dynamic Time Warping, and Feature-Based SVM. There are drawbacks more or less. Some of them can not attain a high accuracy. Others might be time-consuming. In this paper, we proposed a new recognition framework DBCNN for sequences with multi-channels. Our DBCNN framework would pre-process sequences with multi-channels to form "data-bands" and then train a specially designed deep convolutional neural network that uses "data-bands" as inputs. Experiments show that our novel framework could attain a better recognition accuracy

doi:10.1016/j.procs.2015.07.315

compared with other state-of-art methods. Due to characteristics of deep convolutional neural network, training process can be accelerated on GPU in extremely short time, which makes our framework to win more scores.

The paper is organized as follows. Section 2 introduces some backgrounds, including problem description, related works and some basic knowledge about Convolutional Neural Network in brief. Section 3 presents our framework DBCNN, including data processing and deep Convolutional Neural Network design. Section 4 takes a glance at some details of parameters learning. Section 5 reports experiment results of our model comparing with two other state-of-art methods. And Section 6 concludes this paper.

2 Background

2.1 Problem Description

Now we describe multi-channels' sequence recognition task more formally : multi-channels' sequence recognition is a classification task with $\mathcal{N}$ categories, and each of these categories has a corresponding training set $\mathcal{D}_i$. The total training set is $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_{\mathcal{N}}\}$. Training set $\mathcal{D}_i$ has $\mathcal{N}_i$ action instances: $\mathcal{D}_i = \{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_{\mathcal{N}_i}\}$. As mentioned before, each motion instance is constituted by several sequences gathered by different sensors, so we have $\mathcal{M}_i = \{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_{channel\_num}\}$. Assuming that processed sequences have t dimension, sequence $\mathcal{S}_i$ would be a vector with length t. The motion recognition task is to build a classifier given training dataset $\mathcal{D}$ and their corresponding labels.

2.2 Related Works

A wide variety of sequence recognition models have been proposed, but nearly all of them are just focus on specific task, such as motion sequence recognition, musical sequence recognition and so on. Because our contrast experiments are conducted on an open motion sequence dataset, we adopted several motion sequence recognition modesl as our baseline models, such as Hidden Markov Models [6, 3], Dynamic Time Warping [5], and Feature-Based Support Vector Machine [8] etc. Among models of HMMs, continuous-HMM proposed in [6] claimed to gain the highest accuracy. In a task with 10 pre-setting motions, it got a classification accuracy of 96.76%. This result has beaten the discrete-HMM [3], which could achieve an accuracy of 96.1000% in the same dataset. Liu et al. has given a better result than both discrete-HMM and continuous-HMM in [5] using Dynamic Time Warping. SVM could not be used in our task directly due to the large scale and high dimensionality of sequences. Wu et al. [8] proposed a feature-based SVM algorithm, which extracts new features with a lower dimension from the original sequences in a manual procedure. Wang et al. [7] proposed a similar method in different feature extraction rules. Experiment results in [7] show that Wang's method could achieve higher accuracy comparing with HMMs, DTW and Featured-SVM designed by Wu, so we adopt this Feature-SVM model as one of our baseline methods in our experiment.

2.3 Introduction to Convolutional Neural Network

2.3.1 Convolutional Layer in CNN

Although full connection structure in multi-layer perception could make the networks to have more ability in expression, model will be more possible to overfit the data. CNN solves these problems via sparse connection structure, as shown in Fig. 1. First, nodes in high layer only

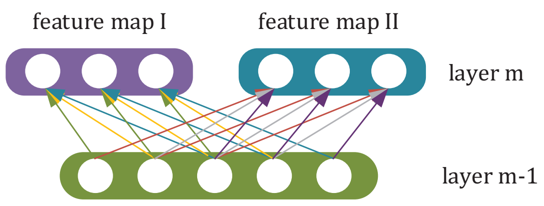

Figure 1: Shared weights of Convolutional Layers in CNN

connect to parts of nodes in low layer, which could reduce the number of parameters efficiently. Moreover, nodes in high layer share identical weights. Just like Fig. 1, arrows with the same weight value have identical colors. This structure make the model to detect position independent features, from which we benefit a lot.

2.3.2 Sub-sampling Layer in CNN

Pooling is a form of non-linear down-sampling in CNN. Among several different pooling methods, max-pooling and average-pooling are the most common. LeCun et al. [4] used this method in their LeNet-5 for two reasons: reducing the computational complexity for upper layers and providing a form of translation invariance.

3 the DBCNN Framework for Multi-Channels' Sequence Recognition

3.1 Data Normalization and Data Augmentation

Generally, different sequences might have different duration time. However, sequences gathered by intelligent devices are usually sampled in a constant time interval. So our obtained sequences instances would have different dimensionality. To overcome this, our model introduces an approximation, linear interpolation. We get values of a sequence in specific proportional position to form new feature vectors, which have a fixed dimensionality. Actually, the value in specific proportional position is a linear combination of two real sample values near the position.

The strong learning capacity of deep convolutional neural network might lead to overfitting, especially when training instances are not enough. Then the model would get a relatively nice result in training set but with a terrible one in test set. To avoid this situation, we bring in the data augmentation idea into our framework. Assuming that we want to get instances $\mathcal{M}_i = \{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_{channel\_num}\}$ in which sub-sequence $\mathcal{S}_i$ is a vector with a fixed length t, we would use data normalization mentioned before to get a $(t+N-1)$-d vector instead of a t-d vector. Then each t-d sub-vector in the $(t+N-1)$-d vector could form a new sub-sequence for training. Now we could get N multi-channels' sequences from just one original multi-channels' sequences. So data augmentation could increase the amount of training instances effectively. It is worth to notice that we would not bring in data augmentation for our test instances. That means we would use data normalization method to construct sequences, in which each sub-sequence is just a t-d vector.

3.2 Data-Band

Now, the new sequence instances are in a same dimension. In our experiment, each instance consists of six channels. Instead of arranging the six sub-sequences into a long vector, we

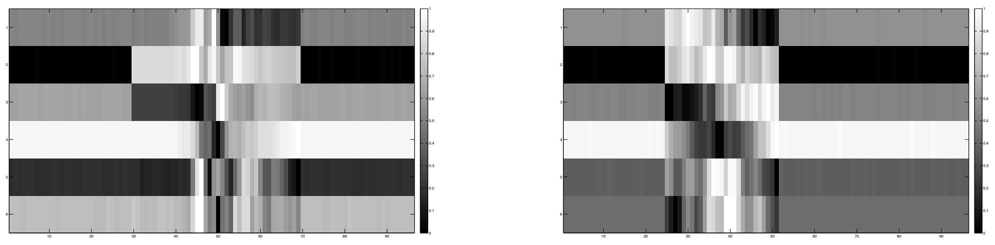
Figure 2: Data bands of two multi-channels' sequences (motion sequences) belong to the same class

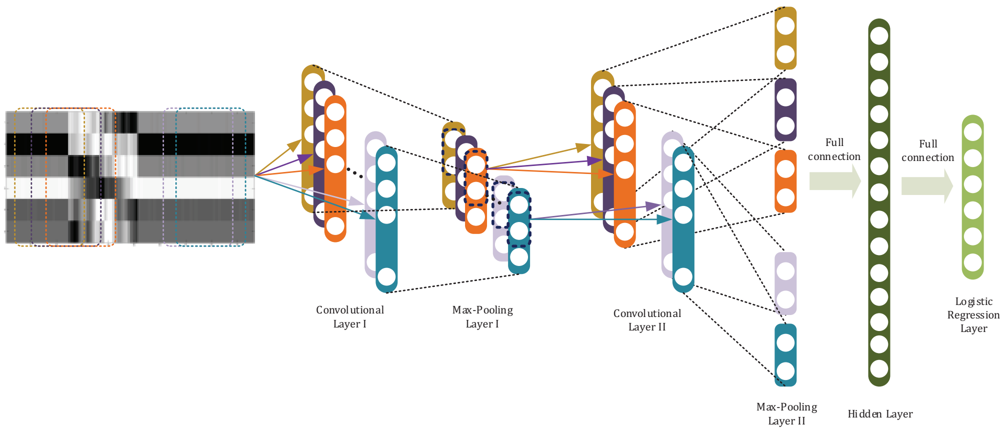

Figure 3: the DBCNN framework for multi-channels' sequences recognition

organize them into a "data-band", which is a narrow matrix with a small row number and a large column number, just like a band.

Fig. 2 shows "data-bands" of two motion sequence instances belong to a same category. It is obvious that they have position invariance: there are similar parts appearing in distinct positions. Our model use "data-bands" as input and could grasp the position invariant successfully, which could not be achieved if we organize the data into a long vector.

3.3 Specially Designed CNN for Recognition of Motion Sequence

3.3.1 Input of CNN

In our proposed CNN for multi-channels' sequence recognition, the input of network is actually the "data-bands" mentioned before. In Fig. 3, the input of the network is showed in the left side clearly. These "data bands" are matrices with dimensionality of $n_{channel} \times t$. The rows of matrices correspond to $n_{channel}$ channels in one instance and the columns correspond to t sampled points, as mentioned in Section 3.1 and Section 3.2.

3.3.2 Design of Convolutional Layer

LeCun et al. [4] have shown that CNN model is extremely suitable for data with high dimensionality and invariant attributes. Sequences of motion instance just fit the properties mentioned above. However, in LeNet-5, receptive fields are little squares to capture the invariance between images pixels. In our model, we treat our pre-processed "data-bands" as images. The difference is that the invariance of "data-bands" exhibits in a narrow form spreading in the lengthways dimension in "data-bands" matrices. If we use small squares as receptive fields, more receptive fields are necessary to catch the invariance. At the same time, the CNN model has more parameters and is more possible to be overfitting. Our model solves the invariance extraction problem in a very elegant way: we set up several convolutional windows to cover the lengthways of "data bands" matrices, just like dashed rectangles with distinct colors in left of Fig. 3. This design could catch the invariance in lengthways using relatively small number of receptive fields. Moreover, each convolutional window scans a "data band" and could get a 1-D feature vector, not a matrix in general CNN for image recognition.

3.3.3 Pooling Layer

A pooling layer provides a further dimensionality reduction. We concatenate a max-pooling layer after a convolutional layer to shrink the feature space. At the same time, pooling layer provides an auxiliary way to grasp position invariant feature.

3.3.4 Classifier

After two convolution-pooling structures, we connect a hidden layer and a logistic regression layer. They form a general multi-layer perception. In fact, two convolution-pooling layers mentioned before could be regarded as a function of feature transformation.

4 Details of Learning

4.1 Gradients in Convolutional Layers

In convolutional layers, previous layer's feature maps are convolved with kernels, then activation function would act on that value to form the output feature maps.

$$\boldsymbol{x}_j^l = sigmoid(\boldsymbol{u}_j^l) \tag{1}$$

$$\boldsymbol{u}_j^i = \sum_{i \in M_j} \boldsymbol{x}_i^{l-1} * \boldsymbol{k}_{ij}^l + b_j^l \tag{2}$$

In Eq. 2, M_j means the selection of input maps in layer $l-1$.

Sensitivity computation process in convolutional layers and sub-sampling layers is not the same with that one in general neural network:

$$\boldsymbol{\delta}_j^l = \beta_j^{l+1}(f'(\boldsymbol{u}_j^l) \circ up(\boldsymbol{\delta}_j^{l+1})) \tag{3}$$

And $up(\cdot)$ function is exactly an inverse process of the down-sampling (e.g. sub-sampling). After we get the sensitivities for maps in convolutional layer, we could compute the gradient as follows:

$$\frac{\partial E}{\partial \boldsymbol{k}_{ij}^l} = \sum_{u,v} (\boldsymbol{\delta}_j^l)(\boldsymbol{p}_i^{l-1})_{uv} \tag{4}$$

$$\frac{\partial E}{\partial b_j^l} = \sum_{u,v} (\boldsymbol{\delta}_j^l)_{uv} \tag{5}$$

We introduce a new variable $(\boldsymbol{p}_i^{l-1})_{uv}$ here, which means the patch in $\boldsymbol{x}_i^{l-1}$ that was multiplied elementwise by $\boldsymbol{k}_{ij}^l$ during the computation of element at (u, v) in the output convolution map $\boldsymbol{x}_j^l$.

4.2 Gradients in Sub-sampling Layers

Sub-sampling layers don't change the number of input maps, and they are really downsampled versions which will downsize the output maps as follows:

$$\boldsymbol{x}_j^l = f(\beta_j^l down(\boldsymbol{x}_j^{l-1}) + b_j^l) \tag{6}$$

Here, $down(\cdot)$ means the sub-sampling function, such as average-pooling or max-pooling. To compute the gradients about β_j^l and b_j^l, we adopt similar steps to compute the sensitivities and gradients.

$$\boldsymbol{\delta}_j^l = f'(\boldsymbol{u}_j^l) \circ \boldsymbol{q}_j^l \tag{7}$$

In Eq.7, $\boldsymbol{q}_j^l$ is a matrix which has the same size of $\boldsymbol{\delta}_j^l$, and its element (u, v) is the convolution between all sensitivities of nodes in layer $l + 1$ that have connection with the node (u, v) in layer l and the weights defines as $\boldsymbol{k}_j^l$. Now, we could compute the gradients for b and β as the following equations:

$$\frac{\partial E}{\partial b_j} = \sum_{u,v} (\boldsymbol{\delta}_j^l)_{uv} \tag{8}$$

$$\boldsymbol{d}_j^l = down(\boldsymbol{x}_j^{l-1}) \tag{9}$$

$$\frac{\partial E}{\partial \beta_j^l} = \sum_{u,v} (\boldsymbol{\delta}_j^l \circ \boldsymbol{d}_j^l)_{uv} \tag{10}$$

After combining all the details mentioned before, we have our parameters optimization algorithm for CNN in **Algorithm 1**.

Algorithm 1 Parameters Optimization for Convolutional Neural Network

```
Require: processed dataset $\mathcal{D}$; learning rate $\alpha$; epoch number $\mathcal{N}$; batch size $n_b$; Network
  Structure $\mathcal{M}$
Ensure: CNN's parameters $\boldsymbol{W}, \boldsymbol{k}, \boldsymbol{b}, \boldsymbol{\beta}$
  initialize network parameters $\boldsymbol{W}, \boldsymbol{k}, \boldsymbol{b}, \boldsymbol{\beta}$ randomly
  for each $i \in [1, \mathcal{N}]$ do
    for each batch data $\mathcal{D}_i$ with size $n_b$ do
      for layer $l := L$ to 1 do
        1. compute sensitivities of layer $l$ according to back-propagation
        2. compute gradient of each parameters in layer $l$
        3. update parameters in layer $l$ in gradient descent method with learning rate $\alpha$
      end for
    end for
  end for
```

Table 1: DBCNN's results for Motion Sequence Recognition

#Kernels	1st Conv Filter	1st Pooling	2nd Conv Filter	2nd Pooling	Accuracy
$[20, 40]$	25×6	3×1	5×1	2×1	98.06%
$[30, 40]$	25×6	3×1	5×1	2×1	98.22%
$[30, 50]$	25×6	3×1	5×1	2×1	98.40%
$[40, 40]$	25×6	3×1	5×1	2×1	**98.66%**
	33×6	4×1	5×1	2×1	98.57%
$[40, 50]$	25×6	3×1	5×1	2×1	98.58%
$[45, 35]$	25×6	3×1	5×1	2×1	98.42%
$[45, 45]$	25×6	3×1	5×1	2×1	98.52%

5 Experiment Results

5.1 Introduction to 6DMG

Our experiments are based on the dataset 6DMG [1], which is a 6-D Motion multi-channels' sequence database developed by Chen et al. in Georgia Institute of Technology. In our experiments, we use parts of original channels: acceleration channels (including $x, y, z-$axies) and angular speed channels (including $x, y, z-$axies). There are 20 kinds of motion in 6DMG. The size of training set, validation set and test set is 32300, 4000 and 5100 respectively.

5.2 Feature-SVM

Feature-SVM proposed by Wang et al. [7] is claimed to outperform HMMs [6, 3], DTW [5] and another featured-SVM given by Wu et al. [8]. We adopt this model as one of baseline experiments. After feature transformation, new features have the length of 48. The details about feature design rules could be found in [7].

New 48-d features are trained using Gaussian Kernel SVM and Linear Kernel SVM with a 5-fold cross validation. For Linear Kernel SVM, the best accuracy 96.9125% is attained when the cost parameter log_2C is 2.3206. And for Gaussian Kernel SVM, the best result appears when log_2C is 1.9459 and log_2g is -2.8332, and the best classification accuracy is 93.7011%.

5.3 Deep Belief Network

Hinton et al. [2] proposed a fast learning algorithm for deep belief network, which use a greedy layer-wise training method. Deep architecture could be built with blocks (e.g. restricted boltzmann machine). Parameters in DBN would be fine-tuned finally.

We get the best result with accuracy of 97.9571% using the DBN model. The structure with the highest classification accuracy has 3 hidden layers, and each layer has 600 nodes. Learning rate in the process is 0.03 with 20% attenuation every 500 epochs.

5.4 the DBCNN model

Table. 1 gives the classification accuracy under distinct parameters configuration. As we could see, our DBCNN framework has achieved a recognition accuracy of 98.66%, which outperforms the other two methods. Left part of Fig. 4 gives a results comparison among these three models.

In our experiments, a GTX-660 graphic unit with 2G GPU RAM is used. Training time using DBCNN model is about 22 minutes on GPU. As a contrast, it takes about 3 times longer

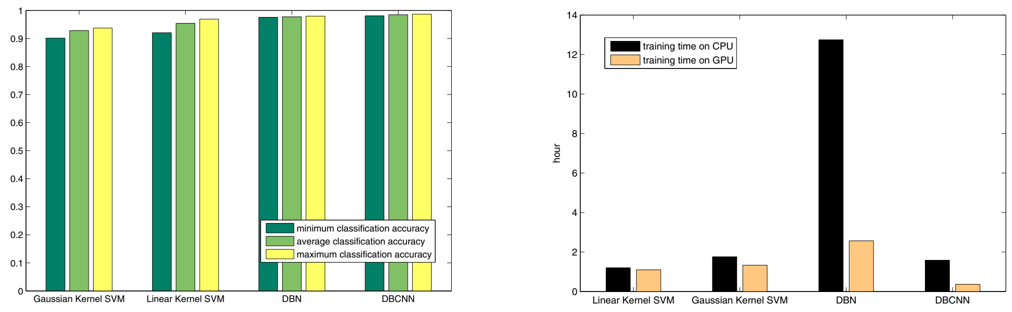

Figure 4: Recognition accuracy comparison and Training time comparison

to train the Feature-SVM model on GPU, and about 7 times longer to train the DBN model on GPU. Training Feature-SVM model and DBN model on CPU might take much longer time. Right part of Fig. 4 shows an approximate training time of these models on CPU and GPU.

6 Conclusion

In this paper, we proposed a novel multi-channels' sequence recognition framework, DBCNN. First of all, we conduct data normalization and data augmentation on raw sequences to form "data-bands". Then we train a specially designed convolutional neural network using these "data-bands" as input. Our experiments demonstrate that our framework could get a higher recognition accuracy than state-of-art algorithms in less training time.

References

[1] Mingyu Chen, Ghassan AlRegib, and Biing-Hwang Juang. A new 6d motion gesture database and the benchmark results of feature-based statistical recognition. In *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on*, pages 131–134. IEEE, 2012.

[2] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[3] Sanna Kallio, Juha Kela, and Jani Mäntyjärvi. Online gesture recognition system for mobile interaction. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 3, pages 2070–2076. IEEE, 2003.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[6] Timo Pylvänäinen. Accelerometer based gesture recognition using continuous hmms. In *Pattern Recognition and Image Analysis*, pages 639–646. Springer, 2005.

[7] Jeen-Shing Wang and Fang-Chen Chuang. An accelerometer-based digital pen with a trajectory recognition algorithm for handwritten digit and gesture recognition. *Industrial Electronics, IEEE Transactions on*, 59(7):2998–3007, 2012.

[8] Jiahui Wu, Gang Pan, Daqing Zhang, Guande Qi, and Shijian Li. Gesture recognition with a 3-d accelerometer. In *Ubiquitous intelligence and computing*, pages 25–38. Springer, 2009.