SenTenCE: A multi-sensor data compression framework using tensor decompositions for human activity classification

Vinay Uday Prabhu
UnifyID
San Francisco CA 94107
`vinay@unify.id`

John Whaley
UnifyID
San Francisco CA 94107
`john@unify.id`

Abstract

In this paper, we introduce SenTenCE (Sensor-Tensor Compression Engine), a tensor decomposition based lossy data compression framework for on-phone Inertial Measurement Unit (IMU) sensor data. We show that this technique achieves impressive data compression ratios with acceptably low percent-root mean square distortion that in turn results in negligible change in the classification accuracy of the Deep Convolutional Neural Network (DCNN) based supervised human activity classifier that is trained with the compressed data instead of the raw uncompressed data.

1 Introduction

Tri-axial Inertial Measurement Unit - Micro Electro-Mechanical Systems (IMU-MEMS) sensors such as accelerometers,gyroscopes and magnetometers are becoming increasingly ubiquitous in modern day commercial smart-phones (and smart-watches), paving the way for some exciting applications spanning domains such as real-time health monitoring, precise indoor navigation, passive seamless authentication and gaming (Refer to [1] for a detailed survey). These on-phone IMU-MEMS sensors also allow for highly accurate Human Activity Classification (HAC) which can then be harnessed to build a finer grained understanding of the activity profile of the smart-phone user [2].
In this regard, there have been several shallow and deep learning approaches proposed [3], that typically entail a *data-collection phase* involving a cohort of volunteers performing a set of predefined activities, followed by a *model-building-and-training phase* and the final *model deployment phase*, where the trained model is deployed either in the cloud or on the device.
Motivated by the dramatic success that Deep Convolutional Neural Networks (DCNN) have achieved in disparate domains such as speech recognition, visual object recognition and object detection [4], researchers in the field of HAC are increasingly embracing DCNN based machine learning solutions [5] replacing the traditional hand-crafted feature engineering driven shallow machine learning approaches [6]. Beyond the attractiveness of circumventing hand-engineering the features, these DCNN based approaches are also more robust to noise [7] which bodes well for the HAC solutions that will be deployed on mainstream commercial hardware. The other rather intriguing aspect of the DCNN approach is that certain architectures such as the MNIST-DCNN architecture[1] ([8]) are remarkably robust to the specific problem domain and require minimal tweaking to achieve acceptable accuracy levels. In fact, in this paper we use that very MNIST-DCNN architecture with minimal customization (see Figure 2) for the HAC dataset ([9]) considered.
Now, before we train this DCNN, capturing the raw sensor data during the data collection phase turns out to be a challenging endeavor. This is on account of issues such as restricted on-phone/on-watch memory capabilities, prohibitively large power consumption and security issues such as

[1]`https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py`

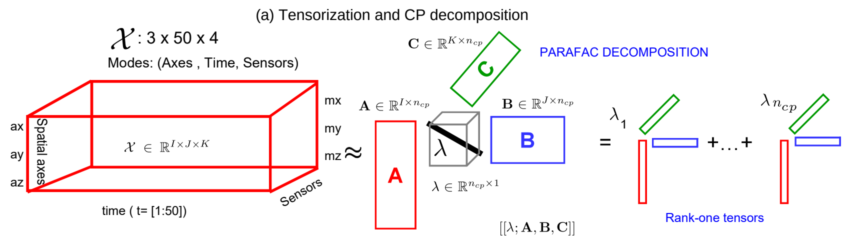

(a) Tensorization and CP decomposition
$\mathcal{X}$: 3 x 50 x 4
Modes: (Axes , Time, Sensors)
ax
ay
az
Spatial axes
$\mathcal{X} \in \mathbb{R}^{I \times J \times K}$
mx
my
mz
time (t= [1:50])
Sensors
$\approx$
$\mathbf{C} \in \mathbb{R}^{K \times n_{cp}}$
C
$\mathbf{A} \in \mathbb{R}^{I \times n_{cp}}$
A
λ
$\lambda \in \mathbb{R}^{n_{cp} \times 1}$
$\mathbf{B} \in \mathbb{R}^{J \times n_{cp}}$
B
$[[\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C}]]$
PARAFAC DECOMPOSITION
=
λ_1
+...+
$\lambda_{n_{cp}}$
Rank-one tensors

Figure 1: *Tensorization*

device fingerprinting ([10]), all of which strongly motivate the usage of an effective compression scheme. In order to tackle this issue, we propose SenTenCE (Sensor-Tensor Compression Engine), a lossy compression framework that uses tensor decomposition or specifically the Canonical Polyadic Decomposition (CPD), also called PARAllel FACtor (PARAFAC) analysis, to compress the sensor data.

The main takeaway point is that that this technique results in impressive compression ratios ($\sim 62\%$) with acceptably low percent-root mean square distortion ($\sim 5\%$) and negligible lowering of the classification accuracy ($< 2\%$) when the compressed signals are *fed through* the DCNN classifier.

The rest of the paper is organized into the following sections. In section 2, we describe the IMU-MEMS dataset used and explain the tensorizing of the dataset. In section 3, we outline the procedure followed to obtain the results of section 4. We conclude the paper in section 5.

2 Dataset description and tensorizing the data

In this paper, we use the HAC dataset introduced in [9]. This dataset consists of 12 IMU-MEMS sensor measurements ; 3 axis Accelerometer(Acc), 3 axis Linear Acceleration Sensor (LinAcc), 3 axis Gyroscope (Gyro) and 3 axis Magnetometer(Mag) mined at a sampling rate of 50 Hz from 10 male volunteers (aged 25-30), via 5 Samsung Galaxy SII (i9100) smartphones placed at 5 body positions. During the data phase, the volunteers were asked to perform 7 physical activities (walking, sitting, standing, jogging, biking, walking upstairs and walking downstairs) which serve as human activity class labels. In this paper, we focus on data emanating from the wrist (from a simulated smart-watch). Given that each of the 10 participants had to perform the 7 activities for 3 minutes each, we have $7 \times 3 \times 60 \times 50 \times 10 = 630 \times 1e^3$ time-measurements of the 12 IMU-MEMS sensor readings in the dataset.

Tensors and tensor decomposition:

Tensors are multidimensional arrays that provide a natural representation for multi-modal data. They are used for feature extraction as well as data compression since they better capture the dependencies in higher-order data-sets compared to matrices.

In [11], multiple ways to arrange Electroencephalogram (EEG) signals into matrices and tensors were explored, and CP tensor decomposition based compression was shown to outperform other matrix and tensor decomposition schemes such as SVD, Column-Row decomposition (CUR), tensor Tucker decomposition and tensor random fiber selection approaches in terms of compression performance.

In our specific scenario with the 12 axis-IMU-MEMS sensor data described above, the 3 modes that naturally emerge are: the 3 spatial axes (X,Y and Z), 4 sensor types (Acc, LinAcc, Gyro and Mag) and lastly, the time indices.

The *tensorizing* of the data is achieved by collecting 1 second of data (50 samples) associated with one specific activity (which results in the matrix $\mathbf{X} \in \mathbb{R}^{50 \times 12}$) and forming a $3 \times 50 \times 4$ tensor, $\mathcal{X}$, as (SeeFigure 1),

$$\mathcal{X}_{sti} = \mathbf{X}_{t,(3i+s)}; s \in \{0,1,2\},\ t \in \{0,..,50\},\ i \in \{0,..,3\}. \tag{1}$$

3 Procedure

Now, given a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, the CPD can be thought of as the generalization of the matrix Singular Value Decomposition (SVD) to tensors (See Figure 1). It decomposes a given tensor [12], into a sum of rank-1 tensors as,

$$\tilde{\mathcal{X}} = \sum_{i=1}^{n_{cp}} [\lambda_i \cdot \mathbf{a}_i \circ \mathbf{b}_i \circ \mathbf{c}_i] \triangleq [[\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C}]]\,, \tag{2}$$

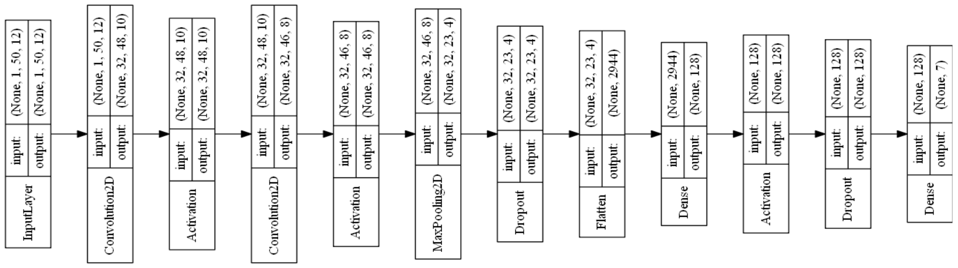


Figure 2: The architecture of the MNIST-DCNN.

such that $\left\|\mathcal{X} - \tilde{\mathcal{X}}\right\|_F^2$ is minimized. Here, $\lambda \in \mathbb{R}^{n_{cp} \times 1}$; $\mathbf{A} \in \mathbb{R}^{I \times n_{cp}}, \mathbf{B} \in \mathbb{R}^{J \times n_{cp}}, \mathbf{C} \in \mathbb{R}^{K \times n_{cp}}$, $\circ$ is the outer (tensor) product operator, $\|.\|_F$ is the Frobenius-norm and n_{cp} is a user-tunable positive integer hyper-parameter.
The SenTenCE framework proposed entails tensorizing the data first (as in (1)), followed by CPD (as in (2)) and finally storing (or transmitting) just the 4-tuple factor matrices $(\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C})$. This results in a compression ratio of,

$$\delta = 100 \left(1 - \frac{(I \times n_{cp} + J \times n_{cp} + K \times n_{cp} + n_{cp})}{I \times J \times K}\right). \tag{3}$$

The tuple $(\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C})$ is used to reconstruct back $\tilde{\mathcal{X}}$ and the quality of the reconstructed signal is measured as the Percent-Root mean square Distortion (PRD), which is defined as, $PRD = \sqrt{\frac{\sum_{i=1}^{3}\sum_{s=1}^{4}\sum_{t=1}^{50}\left(\mathcal{X}_{sti} - \tilde{\mathcal{X}}_{sti}\right)^2}{\sum_{i=1}^{3}\sum_{s=1}^{4}\sum_{t=1}^{50}\left(\mathcal{X}_{sti}^2\right)}} \times 100$. Here, $\mathcal{X}_{sti}$ is the uncompressed IMU-MEMS signal belonging to spatial axis (s), time-index (t) and sensor (i) and $\tilde{\mathcal{X}}_{sti}$ is the compressed signal.
We used the Alternating Least-Squares (ALS) algorithm implemented in the `scikit-tensor` Python module [13] to compute the CP decomposition and compress the sensor data.
The DCNN classifier: As stated in section-I, we used the MNIST-DCNN as shown shown in Figure 2. The input to the DCNN is a 50×12 *image* of raw sensor data (either in compressed or uncompressed form). The model was implemented using `Keras` [8] on a Dell Inspiron 7559 laptop with a 4-core Intel Core i7-6700HQ Processor (2.6GHz),16 GB RAM and Nvidia GTX 960M Maxwell architecture 640 cuda-core graphics processing unit (GPU). We used the $70-10-20$ training -validation - testing split in the experiments.
The code and the dataset have been shared at `https://github.com/vinayprabhu/SenTenCE` to ensure reproducibility of the results.

4 Results

The compression ratios obtained for $n_{cp} = 4$ and $n_{cp} = 5$ were 51.67% and 61.33% respectively. The mean PRD obtained increased *mildly* from 4.0 to 5.79 when n_{cp} was decreased from 5 to 4. In Figure 3(a), we present the distribution of PRD obtained over the entire dataset. The quality of CPD based compression is also showcased via a time-domain visualization of the compressed and uncompressed accelerometer signals along the 3 spatial axes in Figure 3(b). As for the classification task, we used the classification accuracy as the metric, which is, $acc = \frac{\sum_{s=1}^{N_{test}} I\left[\left[l_s == \hat{l}_s\right]\right]}{N_{test}}$, where l_s is the true label of the s^{th} test sample, $\hat{l}_s$ is the predicted label of the s^{th} test sample and $I\left[\left[l_s == \hat{l}_s\right]\right]$ is the *indicator* function which is 1 when $l_s == \hat{l}_s$ and 0 otherwise.
Figure 4(a) contains the confusion matrix with class-wise accuracies for the 7-class HAC task obtained by the DCNN classifier with compressed data ($n_{cp} = 4$) being used for training as well as testing. Figure 4(b) contains a compilation of classification accuracy figures obtained (in $\%$) when (un)compressed data was used for training and (un)compressed data was used for testing, for varying n_{cp}. As seen, there was a negligible drop in accuracy ($< 2\%$) seen when uncompressed data was replaced by compressed data either during the training phase or the testing phase, a result that strongly validates the usage of the CPD based lossy compression technique being proposed.

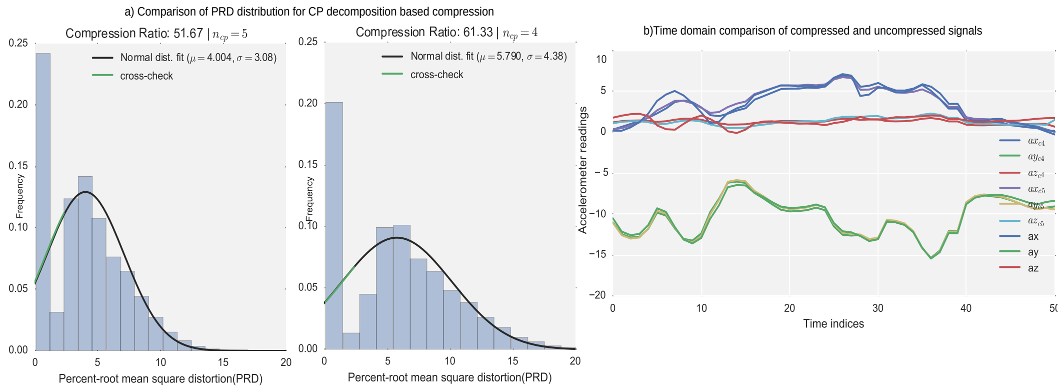
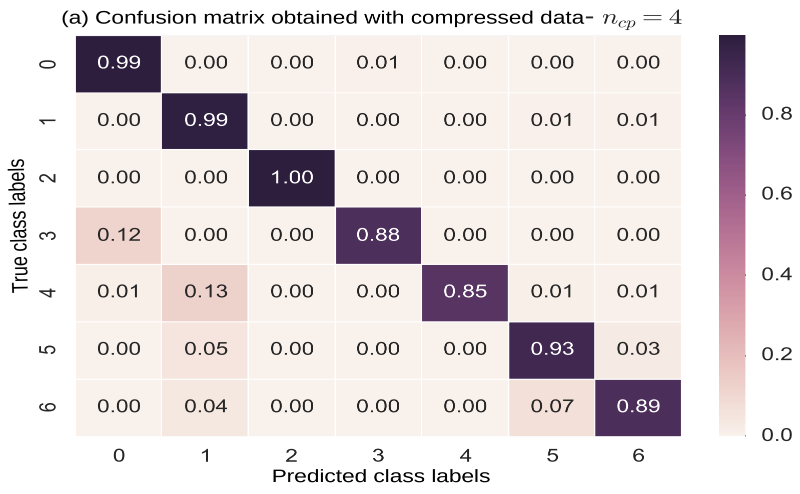


Figure 3: (a): Distribution of PRD for $n_{cp} = 5$ and $n_{cp} = 4$, (b):Comparison of the compressed and uncompressed accelerometer sensor signals in time-domain. Here, the suffix '-c*' in the legend refers to the compressed signal with $* = n_{cp}$ that was chosen.



(b) Classification accuracy variation

Train\ Test	Uncompressed	Compressed($n_{cp} = 5$)	Compressed($n_{cp} = 4$)
Uncompressed	94.28%	93.06%	92.38%
Compressed($n_{cp} = 5$)	93.80%	93.73%	–
Compressed($n_{cp} = 4$)	93.37%	–	93.25%

Figure 4: Classification accuracy results

5 Conclusion and future work

In this paper, we proposed SenTenCE, a CP decomposition based lossy compression framework for on-phone IMU-MEMS sensor data that offered 61% compression ratio while resulting in less than 6% PRD and less than 2% lowering of the classification accuracy of human activity classification using the MNIST-CNN deep classifier. In the near future, we would be replicating the work for varying n_{cp} and across multiple datasets and deep-net architectures. We also would like to investigate the usage of specialized 2D *signal imaging* techniques such as 2D-DFT before feeding the sensor matrix signal into the deep-net classifier. We are also looking to implement CP decomposition on phone and measure the execution time and power consumption metrics for the compression algorithm. Lastly, we will also be exploring disparate tensor decomposition approaches, such as Tucker decomposition [12] and providing comprehensive benchmarking with other approaches such as symbolic representation of time series [14] and piecewise-segmentation based compression [15].

References

[1] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, “A survey of mobile phone sensing,” *IEEE Communications magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[2] J. Zhu, P. Wu, X. Wang, and J. Zhang, “Sensec: Mobile security through passive sensing,” in *Computing, Networking and Communications (ICNC), 2013 International Conference on*. IEEE, 2013, pp. 1128–1133.

[3] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, “A survey of online activity recognition using mobile phones,” *Sensors*, vol. 15, no. 1, pp. 2059–2085, 2015.

[4] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, “Convolutional neural networks for human activity recognition using mobile sensors,” in *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*. IEEE, 2014, pp. 197–205.

[6] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, and J. L. Reyes Ortiz, “A public domain dataset for human activity recognition using smartphones,” in *Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013, pp. 437–442.

[7] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning,” 2016, book in preparation for MIT Press. [Online]. Available: http://www.deeplearningbook.org

[8] F. Chollet, “Keras: Deep learning library for theano and tensorflow,” 2015.

[9] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, “Fusion of smartphone motion sensors for physical activity recognition,” *Sensors*, vol. 14, no. 6, pp. 10 146–10 176, 2014.

[10] A. Das, N. Borisov, and M. Caesar, “Exploring ways to mitigate sensor-based smartphone fingerprinting,” *arXiv preprint arXiv:1503.01874*, 2015.

[11] J. Dauwels, K. Srinivasan, R. M. Ramasubba, and A. Cichocki, “Multi-channel eeg compression based on matrix and tensor decompositions,” in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 629–632.

[12] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[13] M. Nickel, “scikit-tensor,” https://github.com/mnick/scikit-tensor, 2014.

[14] J. Lin, E. Keogh, L. Wei, and S. Lonardi, “Experiencing sax: a novel symbolic representation of time series,” *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[15] T.-c. Fu, “A review on time series data mining,” *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.