# STATISTICAL LEARNING USING HIERARCHICAL MODELING OF PROBABILITY TENSORS

*Magda Amiridi*[†], *Nikos Kargas*[‡], *Nicholas D. Sidiropoulos*[†]

[†] : Department of Electrical and Computer Engineering, University of Virginia, VA, USA
[‡] : Department of Electrical and Computer Engineering, University of Minnesota, MN, USA

## ABSTRACT

**Estimating the joint distribution of data sampled from an unknown distribution is the holy grail for modeling the structure of a dataset and deriving any desired optimal estimator. Leveraging the mere definition of conditional probability, we address the complexity of accurately estimating high-dimensional joint distributions without any assumptions on the underlying structural model by proposing a novel hierarchical learning algorithm for probability mass function (PMF) estimation through parallel local views of a probability tensor. This way the overall problem of estimating a joint distribution is divided into multiple subproblems, all of which are conquered independently by applying regional low-rank non-negative tensor models using the Canonical Polyadic Decomposition (CPD). Using conditioning, such parallelization is possible without losing sight of the full model – which can be reconstructed from the local models and the conditional probabilities. We illustrate the effectiveness and potential of our approach through judicious experiments on real datasets.**

**Index Terms:** tensors, probability, polyadic decomposition, non-parametric estimation, distributed and parallel methods.

## 1. INTRODUCTION

Data analytics has become the center of modern technology as massive amounts of data are generated in a number of complex data-intensive fields such as medicine, social networks, finance, sales, marketing and many more, increasing the demand for tools that can store, detect and summarize the multivariate structure of high-dimensional data. Accurate modeling of statistical information lies at the core of data modeling, and estimating an unknown probability distribution function based on observed data is considered the holy grail of unsupervised learning. Joint probability distribution estimation of a set of random variables $X_1, \ldots, X_N$ is the construction of a distribution estimate $\widehat{P}_{X_1,\ldots,X_N}(i_1, \ldots, i_N)$ using realizations sampled from the true unknown probability distribution $P_{X_1,\ldots,X_N}(i_1, \ldots, i_N)$. Once we know the joint probability mass function (PMF) of a set of discrete random variables, then we can calculate the conditional and joint probability distributions of arbitrary subsets of these variables (e.g., $\mathsf{Pr}(X_n = i_n \mid X_1 = i_1, \ldots, X_{n-1} = i_{n-1}$ ) and manipulate them for a large variety of applications including classification, regression, discriminant and incomplete data analysis as well as assessment of the multimodality, skewness, mean, variance, kurtosis, or any other structure in the distribution of the data [1].

Estimating a joint PMF suffers from the problem that the number of unknown parameters grows exponentially as the number of variables increases (the em curse of dimensionality). Considering 10 binary variables, the number of parameters to be estimated in this case is $2^{10} - 1$, ($-1$ since they should sum up to 1). Tensors (i.e., multi-way arrays) provide a natural representation for such massive multidimensional data [2]. A joint PMF of a set of random variables $X_1, \ldots, X_N$ can be modeled as a probability tensor $\underline{\mathbf{X}}$ where the size of each dimension is equal to the alphabet size $I_1, \ldots, I_N$ of the corresponding variable and the indexed elements represent the probability of the particular realization i.e., $\underline{\mathbf{X}}(i_1, \ldots, i_N) = P_{X_1,\ldots,X_N}(i_1, \ldots, i_N)$. A simple approach for joint PMF estimation is counting the occurrences of the joint variable realizations. However, the corresponding empirical probability tensor will be sparse and highly inaccurate since the probability of almost every realization must decay exponentially in the number of variables.

In this work, we address the complexity of accurately estimating high-dimensional joint distributions by proposing a hierarchical learning algorithm for joint PMF estimation through parallel local views of the probability tensor. In case of continuous random variables, the method requires that we fix a set of non-overlapping intervals that cover the support of each variable. By leveraging the mere definition of conditional probability, we decompose the global problem to many local problems – each modeling a conditional distribution over a local hypercube – and we 'stich' the local conditional models together using a coarse tensor containing the aggregate probabilities of the local hypercubes. This

Email: ma7bx@virginia.edu (M. Amiridi), karga005@umn.edu (N. Kargas), nikos@virginia.edu (N.D. Sidiropoulos). Supported in part by NSF IIS-1704074 and IIS-1447788.

way we handle the global problem in a completely parallel, divide-and-conquer fashion, with each thread processing only a piece of the data. Our method combines the Canonical Polyadic Decomposition (CPD), also known as Parallel Factor Analysis (PARAFAC) [3], [4], and Bayes conditioning to come up with a new hierarchical model of a probability tensor, which *is not* equivalent to a global CPD. Our goal is to build a flexible and intuitive non-parametric distribution estimator over both continuous (which we finely-quantize) and discrete (with large alphabet size) domains. We provide convincing results that corroborate the effectiveness of the proposed method using real data.

## 2. PRELIMINARIES

### 2.1. Canonical Polyadic Decomposition

An $N$-way tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a multidimensional array whose elements are indexed by $N$ indices. Any tensor $\underline{\mathbf{X}}$ can be decomposed as a sum of $F$ rank-1 tensors

$$\underline{\mathbf{X}} = \sum_{f=1}^{F} \boldsymbol{\lambda}(f)\mathbf{A}_1(:,f) \circ \mathbf{A}_2(:,f) \circ \cdots \circ \mathbf{A}_N(:,f),$$

where $\boldsymbol{\lambda} \in \mathbb{R}^F$, $\mathbf{A}_n \in \mathbb{R}^{I_n \times F}$, $n = 1, ..., N$, and $\circ$ denotes the outer product (Fig.1). The number of terms $F$ is bounded, and when $F$ is the smallest number for which such decomposition exists, then $F$ is the rank of $\underline{\mathbf{X}}$, and the decomposition is called *canonical* (polyadic decomposition). A particular element of the tensor is given by $\underline{\mathbf{X}}(i_1, i_2, \ldots, i_N) = \sum_{f=1}^{F} \boldsymbol{\lambda}(f) \prod_{n=1}^{N} \mathbf{A}_n(i_n, f)$. Here, we have assumed without loss of generality that the columns of $\{\mathbf{A}_n\}_{n=1}^N$ have unit norm and have absorbed the scaling into $\boldsymbol{\lambda}$. We use the shorthand notation $\underline{\mathbf{X}} = [\![\boldsymbol{\lambda}, \mathbf{A}_1, \ldots, \mathbf{A}_N]\!]$ to denote the decomposition.

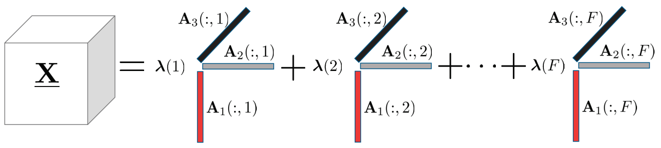

**Fig. 1**. CPD model.

### 2.2. Naive Bayes Model and Rank Decomposition

A joint PMF can always be represented by a latent variable model following the naive Bayes hypothesis [5], [6]. More specifically, the joint PMF of $\{X_n\}_{n=1}^N$ can be expressed as

$$P_{X_1,\ldots,X_N}(i_1,\ldots,i_N) = \sum_{f=1}^{F} P_H(f) \prod_{n=1}^{N} P_{X_n|H}(i_n|f),$$

with $F \leq \min_k \prod_{n=1, n\neq k}^{N} I_n$, where $P_{X_1,\ldots,X_N}(i_1,\ldots,i_N) = \mathsf{Pr}(X_1 = i_1, \ldots, X_N = i_N)$, $P_H(f) = \mathsf{Pr}(H = f)$ is the prior distribution of the latent variable $H$ and $P_{X_n|H}(i_n|f) = \mathsf{Pr}(X_n = i_n|H = f)$ are the conditional distributions [5], [6]. Defining $\mathbf{A}_n(i_n, f) = P_{X_n|H}(i_n|f)$, $\boldsymbol{\lambda}(f) = P_H(f)$, we can easily see that the naive Bayes model can be interpreted as a non-negative polyadic decomposition with the additional constraint $\mathbf{1}^T\boldsymbol{\lambda} = 1$ [7], [8].

## 3. MULTIRESOLUTION CPD

Non-negative CPD models have already been proposed for PMF estimation [9], [6]. However, when dealing with large alphabets (e.g., finely-quantized continuous random variables), sample complexity and computational complexity become major issues. In this work, we explore a very interesting alternative that is made possible by conditioning – and the very definition of conditional probability which preserves the attractive properties of the aforementioned approaches while bringing much needed additional benefits.

### 3.1. Two-layer Approach

For simplicity of exposition, let all random variables take values in $\{1, \ldots, I\}$. We define the following two mappings:

$$\ell(i) := \lceil \frac{i}{L} \rceil \in \left\{1, \ldots, \lceil \frac{I}{L} \rceil \right\},$$
$$r(i) := i - L(\ell(i) - 1) \in \{1, \ldots, L\}.$$

Then $i = L(\ell(i)-1)+r(i)$ and $i \leftrightarrow [\ell(i), r(i)]$ is a bijection. Considering a single random variable $X$, we can write

$$\begin{aligned}\mathsf{Pr}(X = i) &= \mathsf{Pr}(\ell(X) = \ell(i), r(X) = r(i)) \\ &= \mathsf{Pr}(\ell(X) = \ell(i))\mathsf{Pr}(r(X) = r(i) \mid \ell(X) = \ell(i)).\end{aligned}$$

Defining random variables $Y := \ell(X)$ and $Z := r(X)$,

$$P_X(i) = P_Y(\ell(i))P_{Z|Y}(r(i) \mid \ell(i)).$$

Extending to two random variables $X_1, X_2$, we have

$$\begin{aligned}&\mathsf{Pr}(X_1 = i_1, X_2 = i_2) \\ &= \mathsf{Pr}(\ell(X_1) = \ell(i_1), r(X_1) = r(i_1), \ell(X_2) = \ell(i_2), r(X_2) = r(i_2)) \\ &= \mathsf{Pr}(\ell(X_1) = \ell(i_1), \ell(X_2) = \ell(i_2)) \\ &\quad \mathsf{Pr}(r(X_1) = r(i_1), r(X_2) = r(i_2) \mid \ell(X_1) = \ell(i_1), \ell(X_2) = \ell(i_2)).\end{aligned}$$

With obvious notation,

$$\begin{aligned}P_{X_1,X_2}(i_1, i_2) = &P_{Y_1,Y_2}(\ell(i_1), \ell(i_2)) \\ &P_{Z_1,Z_2|Y_1,Y_2}(r(i_1), r(i_2) \mid \ell(i_1), \ell(i_2)).\end{aligned}$$

For three random variables $X_1, X_2, X_3$, we have

$$\begin{aligned}&P_{X_1,X_2,X_3}(i_1, i_2, i_3) = P_{Y_1,Y_2,Y_3}(\ell(i_1), \ell(i_2), \ell(i_3)) \\ &\quad P_{Z_1,Z_2,Z_3|Y_1,Y_2,Y_3}(r(i_1), r(i_2), r(i_3) \mid \ell(i_1), \ell(i_2), \ell(i_3)).\end{aligned}$$

Dropping subscripts on the left hand side, and letting

$$\begin{aligned}&Q(\ell(i_1), \ell(i_2), \ell(i_3)) := P_{Y_1,Y_2,Y_3}(\ell(i_1), \ell(i_2), \ell(i_3)), \\ &S_{\ell(i_1),\ell(i_2),\ell(i_3)}(r(i_1), r(i_2), r(i_3)) := \\ &\quad P_{Z_1,Z_2,Z_3|Y_1,Y_2,Y_3}(r(i_1), r(i_2), r(i_3) \mid \ell(i_1), \ell(i_2), \ell(i_3)),\end{aligned}$$

we have:

$$P(i_1, i_2, i_3) = Q(\ell(i_1), \ell(i_2), \ell(i_3))S_{\ell(i_1),\ell(i_2),\ell(i_3)}(r(i_1), r(i_2), r(i_3)).$$

The above PMF can be interpreted as a decomposition of a coarse, a low-resolution "prior" and a fine PMF from a collection of conditional distributions that resolve a finer level of detail (Fig. 2). The idea now is to build non-negative CPD models for the low-resolution tensor $Q(\cdot,\cdot,\cdot)$ of size $\lceil \frac{I}{L} \rceil^3$ and the refinement tensors $S_{\ell(i_1),\ell(i_2),\ell(i_3)}(\cdot,\cdot,\cdot)$, each of size $L^3$. That is, we shall decompose

$$Q(\ell(i_1),\ell(i_2),\ell(i_3)) = \sum_{f=1}^{F} \boldsymbol{\lambda}(f)\mathbf{A}_1(\ell(i_1),f)\mathbf{A}_2(\ell(i_2),f)\mathbf{A}_3(\ell(i_3),f), \text{ and}$$

$$S_{\ell(i_1),\ell(i_2),\ell(i_3)}(r(i_1),r(i_2),r(i_3)) = \sum_{f=1}^{F^{(g)}} \boldsymbol{\lambda}^{(g)}(f)\mathbf{A}_1^{(g)}(r(i_1),f)\mathbf{A}_2^{(g)}(r(i_2),f)\mathbf{A}_3^{(g)}(r(i_3),f),$$

where $g = g(\ell(i_1),\ell(i_2),\ell(i_3))$, and express the joint PMF $P_{X_1,X_2,X_3}(i_1,i_2,i_3)$, as a product of CPDs of two tensors.

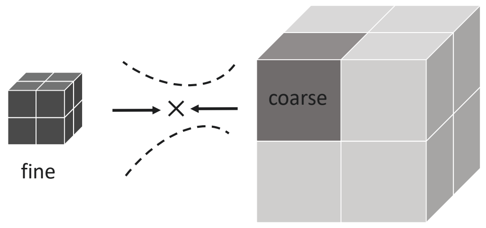

**Fig. 2**. Multiresolution CPD.

The idea naturally generalizes to multiple random variables. Expressing a probability tensor as a product of two (or more) CPDs, as opposed to a single CPD gives us two great advantages: All $\lceil \frac{I}{L} \rceil^3+1$ CPDs (i.e., the $\lceil \frac{I}{L} \rceil^3$ high-resolution conditionals and the one low-resolution prior) can be computed completely in parallel. There is no need for data sharing or communication between different processors. Each tensor is much smaller, which further speeds up computations. This enables massive parallelism. In addition, if one estimates the decompositions directly from data, the data can be split in blocks, and each processor only needs access to points falling in its own block. The low-resolution layer only needs to know the point count in each block. Another advantage of local estimation is the following: considering the distribution to be of high rank, recursive splitting may turn out to be a method to create sub-tensors of lower rank. Smaller size allows using smaller tensor ranks, without sacrificing accuracy.

## 3.2. The Hierarchical Approach

One can consider more than two such decomposition layers. The hierarchical approach extends the above idea by recursive splits of the dimensions of the full probability tensor of the training data. Deciding on an appropriate number of refinement layers (i.e., depth of recursion) is not obvious, considering the shrinking block size, one can quickly end up with too few data samples per block. To address this issue we initially split each dimension $I_1,\ldots,I_n$ in half and assign a binary label (dense or sparse) to each sub-tensor, depending on the relative density of current sub-tensors, (i.e., number of data points of each sub-tensor compared to the average sub-tensor density). On the next layer, only the dense refinement tensors are further split and the average density is updated (Fig. 3). In this way, we avoid overfitting in case of unbalanced data samples. This procedure is recursively repeated until the desired level of refinement is achieved, which can either be manually picked, depending on the number of training samples (more samples yield more layers), or we can let the algorithm reach the maximum depth by itself if the leaf sub-tensors' densities no longer exceed a predefined threshold. In this way we acquire a set of labeled refinement tensors and their corresponding low-resolution tensors.

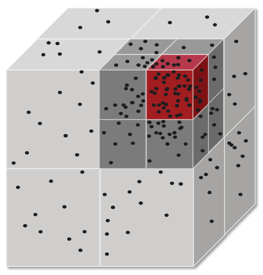

**Fig. 3**. Recursive split.

Different refinement tensors can have different dimensions and different ranks (different from the low-resolution tensor). Because of the heterogeneity of the resulting refinement sub-tensors, another issue that has to be addressed is how to choose the rank of each block efficiently. To build the refinement tensors with the same overall number of parameters as for the full tensor model (i.e., single CPD) approach of rank $F$, for each block we use rank of size $R_{\text{fine}} = \frac{L^2 F}{I^2}$. Naturally, if we have the freedom to assign higher rank to the blocks that are labeled as dense, the approximation of the distribution at that layer is expected to improve.

Each refinement tensor and its corresponding low-resolution tensor is approximated in parallel by a CPD model using Kullback-Leibler (KL) divergence as the fitting criterion since we noticed that KL outperforms Frobenius norm in terms of prediction accuracy in the datasets we experimented with. Defining KL divergence between two probability tensors $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ as

$$D_{KL}(\underline{\mathbf{X}}\|\underline{\mathbf{Y}}) := \sum_{i_1,\ldots,i_N} \underline{\mathbf{X}}(i_1,\ldots,i_N) \log \frac{\underline{\mathbf{X}}(i_1,\ldots,i_N)}{\underline{\mathbf{Y}}(i_1,\ldots,i_N)},$$

we propose solving the following optimization problem for both high- and low-resolution tensors:

| | Binary | | | Multiclass | |
|---|---|---|---|---|---|
| **Method** | **Skin** | **Bank notes** | **Activity** | **Shuttle** | **Older people** |
| SVM | 92.879 | 98.909 | 65.983 | 90.732 | 90.282 |
| Naive Bayes | 92.434 | 87.636 | 70.146 | 90.767 | 91.453 |
| Decision tree | 99.934 | 97.818 | 96.664 | 98.008 | 96.152 |
| Full model | 99.634 | 87.818 | 96.701 | 97.767 | 94.725 |
| Hierarchical | 99.593 | 98.916 | 96.762 | 97.861 | 94.799 |

**Table 1**. Accuracy on UCI and Kaggle datasets.

$$\begin{aligned}\min_{\boldsymbol{\lambda},\mathbf{A}_1,\ldots,\mathbf{A}_N} \; & D_{KL}\Big(\underline{\mathbf{X}}\|[\![\boldsymbol{\lambda},\mathbf{A}_1,\ldots,\mathbf{A}_N]\!]\Big)\\ \text{subject to} \quad & \boldsymbol{\lambda} \geq \mathbf{0},\\ & \mathbf{1}^T\boldsymbol{\lambda} = 1,\\ & \mathbf{A}_n \geq \mathbf{0},\; n = 1\ldots N,\\ & \mathbf{1}^T\mathbf{A}_n = \mathbf{1}^T,\; n = 1,\ldots,N,\end{aligned}$$

employing the EM algorithm as described in [10], [11].

## 4. NUMERICAL TESTS

In this subsection, we evaluate the method of joint distribution estimation on five different machine learning datasets from the UCI machine learning repository [12] and Kaggle [13] to showcase the effectiveness of the proposed joint PMF estimation method and test the PMF estimate for data classification. Three of the selected datasets correspond to binary classification and two to multi-class classification. Since our hierarchical distribution estimation is a PMF-based approach, we finely discretize any continuous features. In view of the fact that the rank $F$ of the joint PMF tensor is not known a priori, an appropriate rank for our model is found by splitting each dataset such that $70\%$ of the data samples is used for training, $10\%$ for validation and $20\%$ for testing. Hierarchical CPD uses rank of $R_{\text{fine}}$ to model the sparser blocks and a rank of $\alpha R_{\text{fine}}$, where $\alpha \in [3, 8]$ to model the dense ones. For each dataset, we run 10 Monte Carlo simulations with randomly partitioned training, validation and test sets, fit models of different ranks and choose the one which maximizes the accuracy over the validation set. After applying the hierarchical CPD approach for estimating the joint PMF, we predict for each data point of the test set the corresponding label using the Maximum a Posteriori (MAP) rule.

| **Datasets** | **N** | **M** |
|---|---|---|
| Skin | 4 | 245057 |
| Bank notes | 5 | 1372 |
| Activity | 9 | 75128 |
| Shuttle | 9 | 58000 |
| Older people | 6 | 100000 |

**Table 2**. Dataset information.

We use 3 different classical classifiers as baselines; linear SVM, a naive Bayes classifier and decision tree using the raw data and additionally we employ PMF estimation using the full CPD model. Table 1 shows the classification accuracy obtained on the datasets and table 2 shows the dataset information. It is noteworthy that in some cases local view captures the essence of the joint distribution even better than the full CPD view. One can see that the performance of our method is always either comparable or superior to the baselines. Nevertheless, we have to consider that our method constitutes a general tool for joint distribution estimation that can model any desired optimal estimator using MAP rule without any additional modeling, given that the distribution is well approximated. For example, our method can be used to predict any other variable from the remaining variables, without training a new model. It can also be used to handle randomly missing variables – the MAP or MMSE estimator is still easy to compute using our model, but none of the other trained methods work in these cases. To illustrate how reliable the joint distribution is, table 3 showcases the KL divergence of training and test set of the full CPD model and the proposed hierarchical model. As expected, PMF learning is refined both in hierarchical and full CPD with growing training data size.

| **Method** | **Skin** | **Bank notes** | **Activity** | **Shuttle** | **Older people** |
|---|---|---|---|---|---|
| H-CPD Train | 1.566 | 4.965 | 1.668 | 0.207 | 0.673 |
| F-CPD Train | 1.642 | 5.628 | 1.834 | 0.291 | 0.777 |
| H-CPD Test | 2.560 | 10.716 | 3.356 | 0.612 | 1.171 |
| F-CPD Test | 2.427 | 12.126 | 3.442 | 0.676 | 1.341 |

**Table 3**. KL divergence on UCI and Kaggle datasets.

## 5. CONCLUSIONS

We proposed a novel method for the fundamental problem of joint PMF estimation. Our method features two main competitive advantages: it enables more accurate distribution estimates faster and at lower complexity due to the parallelism (and lower local rank) it naturally enables. Experimentation on real data has validated the efficacy of the proposed method, which approaches or exceeds the most common custom-designed classification techniques, while affording great flexibility in effortlessly dealing with missing variables or predicting another variable without any model retraining or reconfiguration.

## 6. REFERENCES

[1] C. Fraley and A. E. Raftery, “Model-based clustering, discriminant analysis, and density estimation,” *Journal of the American statistical Association*, vol. 97, no. 458, pp. 611–631, 2002.

[2] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer, “Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis,” *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 71–79, Sept. 2014.

[3] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, “Tensor decomposition for signal processing and machine learning,” *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, July 2017.

[4] R. A. Harshman, “Foundations of the PARAFAC procedure: Models and conditions for an “explanatory” multimodal factor analysis,” *UCLA Working Papers Phonetics*, vol. 16, pp. 1–84, 1970.

[5] N. Kargas and N. D. Sidiropoulos, “Completing a joint PMF from projections: A low-rank coupled tensor factorization approach,” in *2017 Information Theory and Applications Workshop (ITA)*, Feb. 2017, pp. 1–6.

[6] N. Kargas, N. D. Sidiropoulos, and X. Fu, “Tensors, learning, and “Kolmogorov extension” for finite-alphabet random vectors,” *IEEE Transactions on Signal Processing*, vol. 66, no. 18, pp. 4854–4868, Sept. 2018.

[7] A. Shashua and T. Hazan, “Non-negative tensor factorization with applications to statistics and computer vision,” in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 792–799.

[8] L-H Lim and P. Comon, “Nonnegative approximations of nonnegative tensors,” *J. Chemometr.*, vol. 23, no. 7-8, pp. 432–441, July 2009.

[9] D. Lowd and P. Domingos, “Naive bayes models for probability estimation,” in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 529–536.

[10] M. Shashanka, B. Raj, and P. Smaragdis, “Probabilistic latent variable models as nonnegative factorizations,” *Computational intelligence and neuroscience*, 2008.

[11] K. Huang and N. D. Sidiropoulos, “Kullback-Leibler principal component for tensors is not NP-hard,” in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct. 2017, pp. 693–697.

[12] D. Dua and E. K. Taniskidou, “UCI machine learning repository,” 2017.

[13] J-L Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, “Transition-aware human activity recognition using smartphones,” *Neurocomputing*, vol. 171, pp. 754–767, 2016.