Applied Energy xxx (2017) xxx–xxx

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy

A GPU deep learning metaheuristic based model for time series forecasting

Igor M. Coelho [a,b,*], Vitor N. Coelho [a,c,*], Eduardo J. da S. Luz [d], Luiz S. Ochi [c], Frederico G. Guimarães [e], Eyder Rios [f]

[a] Grupo da Causa Humana, Ouro Preto, Brazil
[b] Department of Computing, State University of Rio de Janeiro, Rio de Janeiro, Brazil
[c] Institute of Computing, Universidade Federal Fluminense, Niterói, Brazil
[d] Department of Computing, Universidade Federal de Ouro Preto, Ouro Preto, Brazil
[e] Department of Electrical Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
[f] Institute of Computing, UESPI, Paranaíba, Brazil

HIGHLIGHTS

- A CPU-GPU mechanism is proposed in order to accelerate time series learning.
- Disaggregated household energy demand forecasting is used as case of study.
- Suggestions to embed the proposed low energy GPU based system into smart sensors.
- Parallel forecasting model accuracy evaluation with a metaheuristic training phase.

ARTICLE INFO

Article history:
Received 30 September 2016
Received in revised form 2 January 2017
Accepted 3 January 2017
Available online xxxx

Keywords:
Deep learning
Graphics processing unit
Hybrid forecasting model
Smart sensors
Household electricity demand
Big data time-series

ABSTRACT

As the new generation of smart sensors is evolving towards high sampling acquisitions systems, the amount of information to be handled by learning algorithms has been increasing. The Graphics Processing Unit (GPU) architecture provides a greener alternative with low energy consumption for mining big data, bringing the power of thousands of processing cores into a single chip, thus opening a wide range of possible applications. In this paper (a substantial extension of the short version presented at REM2016 on April 19–21, Maldives [1]), we design a novel parallel strategy for time series learning, in which different parts of the time series are evaluated by different threads. The proposed strategy is inserted inside the core a hybrid metaheuristic model, applied for learning patterns from an important mini/microgrid forecasting problem, the household electricity demand forecasting. The future smart cities will surely rely on distributed energy generation, in which citizens should be aware about how to manage and control their own resources. In this sense, energy disaggregation research will be part of several typical and useful microgrid applications. Computational results show that the proposed GPU learning strategy is scalable as the number of training rounds increases, emerging as a promising deep learning tool to be embedded into smart sensors.

1. Introduction

Sometimes called as the hugest machine ever built, the power grid has been undergoing several improvements. Researchers and the industry have been focusing on efficiently integrating Renewable Energy Resources (RER) into the grid. The massive insertion of RER is usually assisted by Artificial Intelligence (AI) based algorithms and models [2], which are being embedded into Smart Meters (SM) [3]. The proposal described in this current study is a potential tool to be embedded into SM, being able to forecast useful information from big data disaggregated load time series. These load time series have the potential of assisting RER integration in mini/microgrid systems, in which users might employ smart devices to self-manage their resources and demands.

SM are "smart" in the sense that the modest use of sensors is being replaced by devices with plenty of computational abilities.

* Corresponding authors at: Grupo da Causa Humana, Ouro Preto, Brazil.
E-mail addresses: igor.machado@ime.uerj.br (I.M. Coelho), vncoelho@gmail.com (V.N. Coelho).

http://dx.doi.org/10.1016/j.apenergy.2017.01.003
0306-2619/

Usually, these computational abilities are developed based on AI techniques or specific strategies envisioned by its creator/programmer. This class of meters are starting to communicate to each other [4] and to introduce important information to be dealt with by decision makers. These software based sensors are crucial for the decision making process over these scenarios filled with uncertainties.

Among AI techniques found in the literature, deep learning based ones are in evidence. Deep learning has been applied to several classification and regression problems. Part of its success is due to automatic feature extraction at different levels of abstraction. Automatic feature extraction promotes the easy re-utilization of models on different domains without a field-specialist human intervention. Moreover, deep learning allows the representation of the nonlinearities, often associated with complex real-world data. Deep learning models have been used to achieve state-of-the art results in the field of computer vision [5,6] and have also been applied to the problem of time series forecasting [7–10].

Popular deep learning approaches are based on convolutional networks [5], restricted boltzman machines (deep belief networks) [11] and deep autoencoders [12]. However, these methods are often difficult to interpret and reproduce. According to Hu et al. [13] several authors treat deep network architectures as a black box. Another limitation of popular deep learning methods is the high memory consumption [14]. In contrast, the method proposed in this work is of easy interpretation and has low memory consumption, which means a competitive advantage over popular methods of deep learning.

Coelho et al. [15] recently introduced a Hybrid Forecasting Model (HFM), which calibrates its fuzzy rules using metaheuristic based procedures. Without applying any filter or pre-processing energy consumption time series, the HFM model showed to be competitive with other specialized approaches from the literature and easily generalized for performing n-steps-ahead forecast.

The extension proposed here (a substantial extension of the short version presented at REM2016 on April 19–21, Maldives [1]) explores the learning capabilities of the HFM tool, in which feature extraction is done by Neighborhood Structures (NS). Fig. 1 details a generalized version of how the proposed model works. In this current work, only layer 2 is considered, in which the special operator returns the average values of all active functions, namely "activations". NS are used for calibrating each parameter of each activation function: lag input for the backshift operator, rule position and application weight. Furthermore, the metaheuristic calibration algorithm is able to regulate the size of the layer, adding or removing functions.

Motivated by the new class of big data time series, which are reality in several areas (such as in the energy industry, biology, neuroscience, image processing, among others), we decide to enhance the HFM model with a new parallel forecasting evaluation strategy. In particular, in this current work, Graphics Processing Unit (GPU) were designed to be used for forecasting different parts of a microgrid load time series. The use of GPU based architectures can provide a greener alternative with low energy consumption for mining information from such huge datasets [16]. Each GPU provides thousands of processing cores with much faster arithmetic operations than a classic Central Processing Unit (CPU). In a nutshell, we aim at generating ensemble GPU threads learning process, which provide independent forecasts, optimized in order to reduce a given statistical quality measure. GPU seems to fit the scope of the HFM, since the model can be implemented and adapted to GPU computing, particularly because the method uses metaheuristics algorithms and was implemented in the core of the OptFrame [17]. The automatic parameters calibration process of the HFM also matches big data time-series requirements, mainly due to its metaheuristic based learning phase. For this purpose, NS plays a vital role in calibrating the model and finding more efficient solutions. Associated with the power and flexibility of the metaheuristics, the absence of parameters tuning simplify the application of the proposed framework to different times series, in particular, when n-steps-ahead forecasting is required.

This paper considers a mini/microgrid forecasting problem as case of study, the Disaggregated Household Electricity Demand Forecasting. Researchers had begun to publicly release their data sets, such as the Reference Energy Disaggregation Dataset (REDD) [18], which provides low-frequency power measurements (3–4 s intervals) available for 10–25 individually monitored circuits. The household electricity demand forecasting has great potential for microgrid applications, such as the design of green buildings and houses [19]. Forecasting different disaggregated time series from a house opens a wide range of possibilities for efficient RER integration. Considering that billions of dollars are being spent to install SM [20], researchers are advocating that appliance level data can promote numerous benefits.

In the remaining of this paper we introduce the GPU architecture in detail (Section 2) and the GPU disaggregated forecasting process (Section 3). The computational results and the analyzed parameters are presented in Section 4 and, finally, Section 5 draws some final considerations.

2. GPU architecture

The GPU was originally designed for graphic applications (thus receiving the name of a Graphics Processing Unit) such that any non-graphic algorithm designed for GPU had to be written in terms of graphics APIs such as OpenGL. This allowed the development of scalable applications for computationally expensive problems, such as collision simulations in physics [21]. The GPU programming model evolved towards the modern General Purpose GPU (GPGPU), with more user-friendly and mature tools for application development such as CUDA, a proprietary C++ language extension from NVIDIA, one of the main GPU manufacturers [16].

The GPU architecture is organized as an array of highly threaded streaming multiprocessors, each one containing a number of processing units (cores), besides a multi-level memory structure. The configuration of GPU hardware depends on the *compute capability*, a parameter related to GPU micro-architecture that defines which hardware features will be available for CUDA development. A CUDA program consists of one or more phases that are executed in CPU or GPU. The GPU code is implemented as C++ functions known as *kernels*, that are launched by CPU code in a *compute grid*, usually formed by a large number of threads that execute the same kernel aiming at exploiting data parallelism. The threads in a grid are organized in a two-level hierarchy, where first level is arranged as a three-dimensional array of blocks, each one containing up to 1,024 threads. In second level, each block is also arranged in a three-dimensional way. The dimensions of a grid are designed by the programmer and should observe the limits determined by the compute capability of the hardware.

After a kernel is launched, each block is assigned to a single streaming multiprocessors, which executes the threads of a block in groups called *warps*. Each warp is processed according to SIMD (Single Instruction, Multiple Data) model, meaning all threads in a warp execute same instruction at any time. Global memory accesses or arithmetic operations performed by some thread also affects warp execution, forcing all threads in the same warp to wait until the operation is completed. To hide the latency related to those operations, GPU schedules another warp to keep SM busy. This is possible because GPU is able to handle more threads per SM than cores available.

Please cite this article in press as: Coelho IM et al. A GPU deep learning metaheuristic based model for time series forecasting. Appl Energy (2017), http://dx.doi.org/10.1016/j.apenergy.2017.01.003