Using LSTM Neural Networks as Resource Utilization Predictors: The case of training Deep Learning models on the Edge

John Violos[1], Evangelos Psomakelis[2], Dimitrios Danopoulos[1], Stylianos Tsanakas[1], and Theodora Varvarigou[1]

[1] School of Electrical and Computer Engineering, National Technical University of Athens, Zografou, Greece 15771
{violos@mail,dimdano@microlab,el09727@mail,dora@telecom}@ntua.gr
[2] Dept. of Informatics and Telematcs, Harokopio University of Athens, Tavros, Greece
vpsomak@hua.gr

Abstract. Cloud and Fog technologies are steadily gaining momentum and popularity in the research and industry circles. Both communities are wondering about the resource usage. The present work aims to predict the resource usage of a machine learning application in an edge environment, utilizing Raspberry Pies. It investigates various experimental setups and machine learning methods that are acting as benchmarks, allowing us to compare the accuracy of each setup. We propose a prediction model that leverages the time series characteristics of resource utilization employing an LSTM Recurrent Neural Network (LSTM-RNN). To conclude to a close to optimal LSTM-RNN architecture we use a genetic algorithm. For the experimental evaluation we used a real dataset constructed by training a well known model in Raspberry Pies3. The results encourage us for the applicability of our method.

Keywords: Resource Utilization · Edge Computing · Long Short-Term Memory · Deep Learning · Genetic Algorithm.

1 Introduction

Resource and cost optimizations are undoubtedly the hottest issues in the domain of cloud, fog and edge applications and for good reason. The variety of configuration options, cloud providers and deployment architectures, both software and hardware, are giving headaches to the system administrators, causing great losses in money and resources. These resources are especially valuable when we are talking about edge deployments as most edge devices have limited and costly resources. Even if they are interconnected in a cloudlet or IoT clusters their pooled resources are still pretty valuable and should not be wasted. Most users are overestimating their needs, ending up paying for idle resources or limiting the overall system capacity as they are locking unused resources over large periods of time. There is an active interest in the relevant industry in order to

tackle this issue. Amazon, for example, launched a resource optimization service which profiles our usage and makes recommendations to limit the resource waste[1].

The present work is tackling the cost reduction issue by creating a resource prediction model, which is used to optimize the resources and minimize the resource waste while preventing QoS infringements. It is obvious that in cloud and edge deployments there is a direct relation between resources and costs, the higher the allocated resources the higher the cost, even if the resources are not actually used. For that reason we focused on minimizing the waste of resources as a means of reducing the costs. The proposed LSTM-RNN based predictor is using a set of auto-optimized neural networks in order to profile the deployed applications and predict the resource utilization in the near future (8 min time step). This enables us to choose the optimal configuration for our cloud or fog deployment.

The testing and evaluation process took place using an experimental dataset created by Raspberry Pi machines. The Raspberry Pi is a common edge device chosen for its low price, its versatile, multi-purpose nature, its portability and its efficiency in power consumption as well as its low heat generation. The sample application tested was the training of a movie review classifier using Keras over Tensorflow on an IMDB database of movie reviews. This process created a realistic workload in the Raspberry Pi that was monitored using a Python script created specifically for the present work. The monitoring tool gathered information about the CPU, RAM, HDD and Network utilization at regular intervals, creating a training dataset for our algorithms.

The key contributions of this paper are:

- The proposal of a Time-Series Multi-Output Regression model for resource utilization forecasting of multiple resources in a unified model.
- An applicability study of genetic algorithms in the hypothesis space of LSTM-RNN topologies in the domain of resource utilization forecasting.
- The demonstration of an edge deployed AI application profiling approach using readily available monitoring libraries while focusing on key metrics.

The rest of the paper is structured as follows: Section2 briefly reviews the literature about resource utilization forecasting techniques and AI applications deployed on the edge. Section 3 provides a description of the proposed AI application, its deployment environment, and how the profiling was conducted. Section 4 presents the design process of an LSTM-RNN model using the genetic algorithm. Section 5 describes the experimental evaluation and provides an interpretation of the results. Finally, in Section 6 the derived conclusion and future work are described.

2 Related Work

Resource utilization and cost are two tightly interconnected terms in edge computing. There are plenty of ways that resources affect the system costs but the

most referenced one is the energy consumption. For example, Li et. al. [2] are describing a system that tries to minimize the energy consumption of machine learning inference on Unmanned Aerial Vehicle (UAV) systems using a pipeline of a successive convex approximation based algorithm and the Dinkelbach algorithm. In a more close scenario as the one examined in the present, Guo et. al. [3] are measuring the performance, and effectively the cost, of a mobile edge computing (MEC) network based on three factors: a) its energy consumption, b) its latency and c) its system total charge introduced by nearby computational access points. Their approach is purely mathematical, using formulas to describe and predict the three basic values for each network and application.

The main approaches of modeling the resource utilization include analytical models and machine learning models. Analytical models often use queuing theory [4] to model computing infrastructures that operate under steady request arrival rate, average service time and resource utilization with probabilistic behaviour. In addition, new models have been proposed which enhance the queuing models focusing on parallelizable tasks [5].

Machine learning models predict the resource utilization using a data-driven approach. They employ a large number of methods and techniques to discover patterns of resource usage in various circumstances from historical data. An ensemble approach [6] has been proposed integrating a multi-regression model feature selection mechanism. Sequencing resource observations is commonly exploited by time series approaches like Autoregressive Integrated Moving Average models [7]. Furthermore, a research close to our proposed model predicts CPU utilization using evolutionary neural networks [8].

The present work differentiates from the above mentioned models by combining the time series approach with deep learning models. Then it makes a smart search using genetic algorithms to identify an optimal LSTM-RNN that can forecast the resource utilization of CPUs, Ram, Disk I/O, and Network as a Multi-Output Regression Model. The genetic algorithm excels the evolutionary approach in the exploration of hypothesis space as the later use only mutation as reproduction strategy while the former use both crossover and mutation.

3 Application and Deployment Environment

3.1 Description of Applications and Data

The application we selected is a popular machine learning problem, specifically a classification of movie reviews based on the famous IMDB dataset. We trained and deployed our model using Keras, which is a high-level API to build and train models in Tensorflow, achieving almost 90% accuracy in the test dataset. It's worth mentioning that this application was selected as an indicative example which belongs to a larger category of machine learning applications specifically text classification which is a type of algorithm that can be found on the edge .i.e. document translation, language detection, spam detection, etc.

3.2 Description of Deployment Environments

We chose the Raspberry Pies3 as the deployment platform for our application. As for the specifications, this model packs a 64-bit quad-core ARM Cortex-A53 at 1.4GHz running on Raspbian operating system which is basically Debian Linux.

Having a platform that supports Linux based operating systems means we were able to have package support for most of the machine learning libraries and image packages. However, a careful setup was needed in order to install the whole Tensorflow framework on the small Raspberry (installing 3d party libraries, having small RAM, etc.) and run our application. Last, we connected to our device using the SSH protocol in order to control the application remotely (for example pause or resume the training algorithm.

3.3 Parameters of Interest

As a monitoring tool we chose to develop a python script that combines the psutil [9] and GPUtil [10] libraries, providing a plethora of monitoring tools that enable us to register metrics about the CPU, RAM, network, HDD and GPU in real time, while the target processes are running. It is a very lightweight tool, compatible with edge limitations.

The script allows us to define a set of parameters during each runtime, fine-tuning the monitoring process. These parameters include the snapshot frequency of the metrics watched, the file size limit of the logs in order to more easily process the files, and three lists of excluded devices if we have any, one for each category of devices (HDD, GPU and Network). All the data extracted by the monitoring tool are separated by category and converted in JSON format for uniformity and easier usage during the next steps of the modeling process. A JSON schema example is available at the author's GitHub Repository [11]

4 Resource Utilization Predictors Model with LSTM Neural Networks

The resource utilization predictor involves two pipelines as depicted in Fig. 1. The first pipeline is the training process which builds a multi-output regression model based on historic data. Specifically, the Raspberry Pies profile the resource usage during execution of the deployed service. The profiling output contains cpu, ram disk and network parameters. The profiling dataset is pre-processed using the min-max normalization method and piped to the hyper-parameter optimization process which trains multiple neural networks in order to identify an optimal topology.

The second pipeline Fig. 1b starts when the first pipeline has finished. It takes the current status of resource usage, it makes the pre-processing like in the training. Afterwards, the DL model provides a resource utilization prediction for the next 8 min time-step. The resource predictions are of manifold usages. In Fig. 1b, we see how we can leverage these predictions in order to employ elastic scaling.

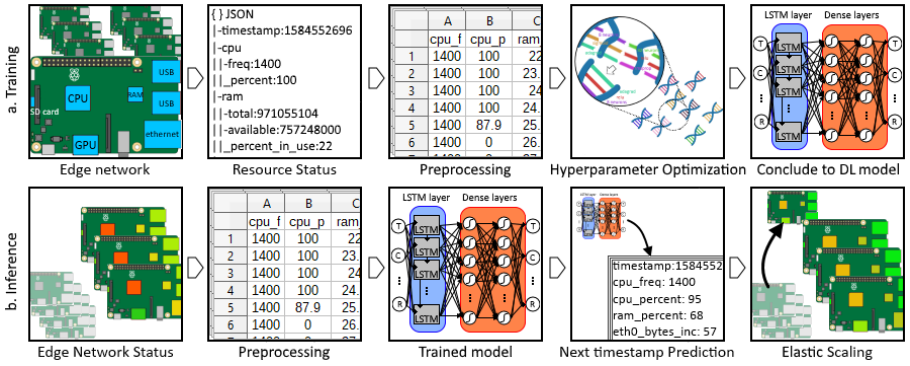

Fig. 1: LSTM-RNN with Genetic Algorithm Training and Inference

4.1 LSTM Neurons Architectural Usage

The training of DL models exhibits predictability in the sequence of resource utilization. We make the assumption that the next CPU and RAM usage may often dependent on the previous values and follows common patterns. LSTM, by having a chain-like structure, are able to connect information from long monitoring periods and map patterns of usage to predictions.

LSTM is a non-linear time series model that allows information to persist, using a memory state, and is capable to learn the order dependence between observations in a sequence. LSTM originates from RNN but outperforms it by resolving two weaknesses: the vanishing gradient problem and the short-term memory. The former makes RNN earlier layers incapable of being trained sufficiently. The latter makes RNN incapable of carrying information from earlier time observations to later ones and as a result RNN forgets faster.

The parameters of an LSTM-RNN can be optimized by minimizing an objective function. Widely used optimization techniques for LSTM-RNN are the Root Mean Square Propagation (RMPSrop) and the Adaptive Moment Estimation (Adam).

4.2 Architecture Design

Hyper-parameters decisions mainly involve the number of layers in an ANN and the number of units in each layer. In addition, it may also include the activation function type and dropout percentage. An automatic, systematic search through architecture space can be performed by a genetic algorithm.

The main steps of Genetic Algorithms are summarized as follows. Firstly, we pick a population size (N) and generate said population of ANN with random hyperparameters. Secondly, we pick a number of generations for the algorithm. We iterate over the generations performing the following steps: i) We train and test the entire population and then sort it by some metric; ii) The (B) best ANN

advance to the next generation along with some randomly chosen (R), while we are being careful to avoid converging too fast; iii) We generate the remaining (N-R-B) members of the population by picking the hyper-parameters of the new networks choosing from the hyper-parameters of two parent ANN, chosen at random* from (B U R) with the probability of a random HP changing to a completely different value (mutation).

5 Experimental Evaluation

The proposed model is implemented and evaluated in Python 3 using the frameworks TensorFlow 2, pandas, NumPy, Scikit-learn, logging and statistics in the Jupyter notebook environment of the Google Colaboratory. The experiments' source code is available for any kind of reproduction and re-examination at the author's GitHub Repository [11].

5.1 Comparison with State of the Art Models

The proposed model is compared against Application and User Context Resource Predictor (AUCROP) model [12], a state of the art predictor for Resource utilization which employs traditional machine learning algorithms, and with two state of the art machine learning meta models named Auto-sklearn and XGBoost.

Auto-sklearn [13] is an automated machine learning pipeline of pre-processing, regression, and hyper-parameter tuning using a Bayesian optimization process. It contains a repository of previous optimization runs enabling training from previous saved settings. XGBoost [14] is a software library that implements the model of Gradient boosted decision trees. It is available in many programming languages including Python and compatible with Scikit-learn. XGBoost has gained much popularity because it has been used by many Kaggle and KDD Cup winning solutions.

5.2 Experimental Results And Discussion

For the evaluation we used the out-of-sample method that preserves the temporal order of records. The dataset was split into two sequences, the training sequence (66% observations) and the testing sequence (34% observations). In the context of time series applications we cannot use any shuffling mechanism as is the common practice for traditional machine learning evaluation, because we must preserve the sequence of observations. For evaluation metrics we used Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). We also log the training time and the Inference Times for a single prediction and for a batch of 100 predictions.

The Genetic Algorithm performed a smart search in hypothesis space in order to converge on an optimal LSTM-RNN Multi-Output Regression model. AUCROP, XGBoost and Auto-sklearn also used a hyper-parameter optimization approach. The list and the candidate values of the hyper-parameters are

described in a file in the author's GitHub Repository [11]. The experimental results are summarized in Table 1.

Table 1: Single-Output & Multi-Output Evaluation

Method	RMSE	MAE	CPU-1 (%)		RAM-1 (%)		Train. Time	Infer. Time	
			RMSE	MAE	RMSE	MAE		Single	Batch
AUCROP	0.0814	0.0414	17.235	14.009	2.480	1.482	522	0,004	0,011
XGBoost	0.1139	0.0599	16.457	13.569	1.515	0.4720	181	0.060	0.010
Auto-sklearn	0.1055	0.0243	52.659	17.856	1.546	0.5260	1338	0.263	0.572
LSTM-RNN	0.0674	0.0338	16.099	12.838	1.746	0.9170	574	0.020	0.024

Looking at the results, the LSTM-RNN model trained using the Genetic Algorithm achieved the best RMSE value over the dataset, while maintaining low training and inference times. In resource utilization prediction the large errors are undesirable, so RMSE is the most important evaluation metric as it assigns relatively high weight to large errors compared to MAE. Table 1 provides aggregations of all regression outputs in the first and second column and specific outputs such as CPU-1 and RAM-1 in the columns three to six.

LSTM-RNN also outperformed all other models in predicting the percentage value of CPU-1 core, measured by both RMSE and MAE. The accuracy of RAM utilization predictions appears to be slightly below XGBoost and Auto-sklearn, possibly due to a randomization factor introduced by RAM's caching policies.

6 Conclusions

The present research demonstrates how a DL approach with LSTM layers can be applied in order to tackle the problem of forecasting the resource utilization in edge devices in the use case of AI model training. LSTM-RNN can be a powerful regression model with high accuracy results if its architecture is of optimal design. A heuristic approach was used to converge on close to optimal architecture using a genetic algorithm.

There are still topics that are out of this paper's scope. Firstly, we should examine different applications deployed on the edge. Applications with erratic workloads should be more interesting. Applications that have fluctuating number of requests should need additional parameters of observations. In this case, we should find different feature engineering techniques and leverage open data related to the deployed service.

The interaction between the edge devices is also a very interesting topic. Edge devices do not work isolated. They belong in a heterogeneous infrastructure involving different types of software and hardware. Resource usage also depends on this ecosystem so predictors should take into consideration the full environment.

Acknowledgements

This work is part of the ACCORDION project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871793".

References

1. "Launch: Resource Optimization Recommendations," Amazon Web Services, Jul. 23, 2019. https://aws.amazon.com/blogs/aws-cost-management/launch-resource-optimization-recommendations/ .
2. M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," IEEE Trans. Veh. Technol., vol. 69, no. 3, pp. 3424–3438, 2020.
3. Y. Guo et al., "Intelligent offloading strategy design for relaying mobile edge computing networks," IEEE Access, vol. 8, pp. 35127–35135, 2020.
4. T. M. Truong, A. Harwood, R. O. Sinnott, and S. Chen, "Performance Analysis of Large-Scale Distributed Stream Processing Systems on the Cloud," in 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), Jul. 2018, pp. 754–761, doi: 10.1109/CLOUD.2018.00103.
5. X. Li, S. Liu, L. Pan, Y. Shi, and X. Meng, "Performance Analysis of Service Clouds Serving Composite Service Application Jobs," in 2018 IEEE International Conference on Web Services (ICWS), Jul. 2018, pp. 227–234, doi: 10.1109/ICWS.2018.00036.
6. G. Kaur, A. Bala, and I. Chana, "An intelligent regressive ensemble approach for predicting resource usage in cloud computing," Journal of Parallel and Distributed Computing, vol. 123, pp. 1–12, Jan. 2019, doi: 10.1016/j.jpdc.2018.08.008.
7. Q. Zia Ullah, S. Hassan, and G. M. Khan, "Adaptive Resource Utilization Prediction System for Infrastructure as a Service Cloud," Computational Intelligence and Neuroscience, Jul. 25, 2017. https://www.hindawi.com/journals/cin/2017/4873459/
8. K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, "Predicting host CPU utilization in the cloud using evolutionary neural networks," Future Generation Computer Systems, vol. 86, pp. 162–173, Sep. 2018, doi: 10.1016/j.future.2018.03.040.
9. GitHub, G. Rodola', giampaolo/psutil, https://github.com/giampaolo/psutil.
10. GitHub, A. K. Mortensen, anderskm/gputil https://github.com/anderskm/gputil.
11. GitHub, S. Tsanakas, STsanakas/Resource_Utilization_Prediction, https://github.com/STsanakas/Resource_Utilization_Prediction.
12. J. Violos, E. Psomakelis, K. Tserpes, F. Aisopos, and T. Varvarigou, "Leveraging User Mobility and Mobile App Services Behavior for Optimal Edge Resource Utilization," in Proceedings of the International Conference on Omni-Layer Intelligent Systems, Crete, Greece, May 2019, pp. 7–12, doi: 10.1145/3312614.3312620.
13. Feurer, M., Klein, A., Eggensperger, K. , Springenberg, J., Blum, M. , and Hutter, F., "Efficient and Robust Automated Machine Learning," in Advances in Neural Information Processing Systems 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970.
14. Chen, T., and Guestrin, C., "XGBoost: A Scalable Tree Boosting System," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.