ISSN 2224-087X. Електроніка та інформаційні технології. 2020. Випуск 13. С. 118–125
Electronics and information technologies. 2020. Issue 13. P. 118–125

УДК 519.85.004.42 DOI: https://doi.org/10.30970/eli.13.12

DEVELOPMENT OF NEURO-CONTROLLER BASED ON STM 32

O. Sinkevych, L. Monastyrskii, Ya. Boyko, B. Sokolovskii

Ivan Franko National University of Lviv,
50 Drahomanova str., Lviv, UA-79005, Ukraine
oleh.sinkevych@lnu.edu.ua

A step-by-step approach for the development of neuro-controller system is considered in this work. In order to utilize STM 32 microcontroller as a hardware solution in the meaning of edge computing we propose to deploy a deep learning model for the tasks of prediction and anomaly detection. The data which are used for the training phase as well as for the inference consist of the smart home indoor temperature time series with the time related patterns. The developed deep learning LSTM model automatically discovers time dependencies during the training and uses the time series features for walk-forward validation. The data preparation stage followed by the re-definition of time series forecasting to supervised learning problem is presented. To calculate a set of the optimal LSTM hyper-parameters the grid search algorithm that implies a step-wise full training neural network process has been applied. Trained network was deployed and tested on STM 32 microcontroller with installed X-CUBE-AI expansion package. The obtained validation and test results show a possibility of using such approach for the development and improvement of the LSTM architectures for edge and mist computing applications.

Key words: edge computing, deep learning, LSTM, smart home, STM 32.

Introduction.

A large variety of IoT applications and solutions in the different industry areas has been recently shifted from the cloud-based systems powered with the high-productivity computational resources (expensive GPU in the case of industry scale deep learning) to the edge computing devices. Autonomous driving systems, swarm intelligence and smart grids show a great division of responsibilities for the decision making between central units and software installed directly on the particular device. Among the rest of advanced technologies, a smart home is of the considerable interest for applying edge and mist computing paradigms [1]. The big advantage of transferring the computational resources closer to the edge is ensuring an ability to keep system components working locally regardless of the internet connection in case of cloud solution. Also, the crucial delays which hurt subtle part of the system can be overcome by using such an approach.

While a typical smart home hub may be deployed on the popular micro-computer like Raspberry Pi 3 [2, 3], a case when the significant part of functional logic is transferred to the controller (mist computing) or some intermediate node opens much wider possibilities for the locally distributed computing. For instance, in [4] authors proposed an algorithm and applied concept of the home automation engine using edge computing. In [5] authors discuss the security and availability issues in smart homes and propose an edge-of-things solution that focuses on putting the management which is controlled by the network operator of the home at the edge.

In this work, we present an initial concept and ideas of smart home mist computing system. Since the sensor data processing is the necessary step for each analytical system it is important to develop and configure a proper algorithm implementation to manage the raw data stream. The examples of such applications are smart thermostats, different alarms, decision making based on resident activity recognition etc. Here we address the problem of anomaly detection and short-term forecasting as the data pre-processing steps where the corresponding implementation is deployed on a STM 32 microcontroller. After pre-processing the cleaned data can be stored in selected database system (SQLite, InfluxDB, MySQL) on the chosen hub (Raspberry Pi 3) or cloud server followed by applying the machine learning algorithms used for further analytics and control. The idea of anomaly detection method utilizes the long short-term memory network implemented directly on a STM 32 via X-CUBE-AI expansion package which supports the deep learning framework TensorFlow Lite.

Neuro-controller concept. To start with, let's introduce a schema of the proposed device. Because the purpose of neuro-controller is to process the data collected from connected sensors installed in a smart home, for the sake of simplicity we will consider only the room temperatures as the input stream. Figure 1 depicts a generic prototype.

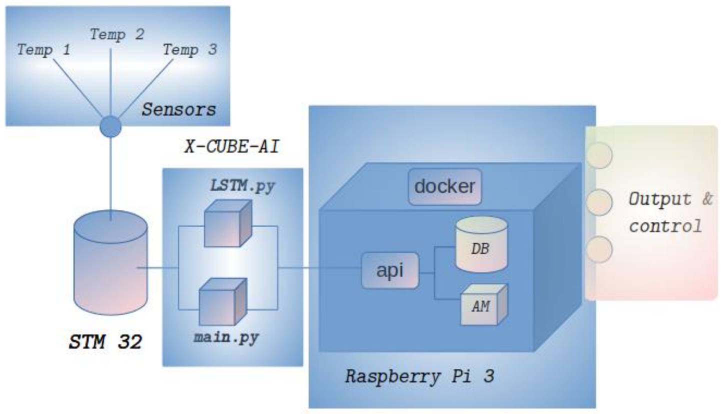

Fig. 1. Neuro-system schema

As it is shown in fig. 1, the room temperature sensors (DHT 11 or DHT 22) are connected to STM 32 microcontroller where the several Python 3 scripts with the implementation of developed neural network are integrated into X-CUBE-AI infrastructure. We have opted Python programming language due to the simplicity of prototyping the models and huge support by the IoT developers community. The detailed description of LSTM (long short-term memory network) will be provided in the next section, but the main goal of this model is to analyze the incoming data and detect anomalous or suspicious time points, report their appearance and replace them using the predicted values.

Next, the processed temperature time series are sent to the Raspberry Pi 3 hub where the docker container with the database system (e.g., InfluxDB) and other programmable controlling units are deployed. The additional modules (AM) can consist of the advanced machine and statistical learning algorithms which provide the high-level analytics of the smart home thermal behavior and heating processes which occur in the building [6]. The output and control units require the additional logic which is not covered in this paper.

Model development.

We assume that the room temperature data $x_r\left(t_1,t_2,\ldots,t_n\right)$ are the time series with some latent time dependencies which can be discovered and used for the predictions and anomaly detection. Among the large number of available techniques to deal with the time series data like linear models (ARIMA), of considerable interest are neural network approaches, namely recurrent networks and their improvements (LSTM, GRU, ConvLSTM, CNN LSTM). In this work we consider the simplest type of LSTM model – univariate Vanilla LSTM with only one hidden LSTM layer. The reasons of such choice are restrictions in the model size and calculations caused by STM 32 system parameters.

To conduct our numerical experiments we have chosen an open-access smart home dataset REFIT which provides a vast set of different smart home measurements [7].

Here, univariate series $\mathbf{x}_r\left(t_1,t_2,\ldots,t_n\right)$ is the single vector of temperature readings during moderately long period of time equal to one season (3 months) with 48 readings per day, totally 4416 values.

Generally, it is recommended to use the normalization or standardization routine to scale input time series in order to preserve gradient exploding and wrong results before training the LSTM. Figures 2, 3 show the summer indoor temperatures and the results of the normality test correspondingly.

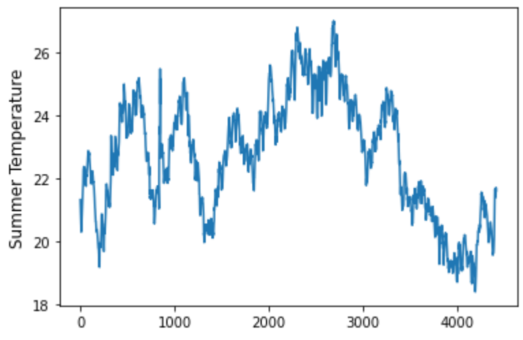

Fig. 2. Summer temperatures

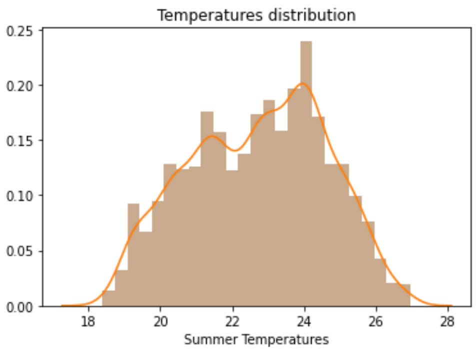

Fig. 3. Summer temperatures distribution

Because the data are not distributed normally, we have chosen the data normalization instead of standardization method. The important step for the data preparation is to check the normalized series for stationarity. If the resulting time series is non-stationary (the presence of trend and seasonality), the difference method should be applied, i.e., $x_{diff}\left(t\right)=x\left(t\right)-x\left(t-diff\_order\right)$, where $diff\_order$ is the order of differencing which can be identified using a grid-search or linear models. In this study the input summer temperatures are stationary time series.

The next step is to convert the input normalized series into the supervised form, i.e., $\mathbf{x}_r\left(t_1,t_2,\ldots,t_n\right)\rightarrow\left(\mathbf{x},y\right)^j$, where $\mathbf{x}$ is the vector of $x\left(t_1\right),x\left(t_2\right),\ldots,x\left(t_l\right)$, l is the number of lags (readings which influence the predicted value y), j is the number of sample. Hyperparameter l is crucial for the precise prediction, so it has to be carefully picked during the investigation of model performance.

The base structure of the model in Keras [8] notation is as follows: a) input layer for $(\mathbf{x}, y)^j$ training set; b) LSTM layer with $num\_nodes \in \mathbb{N}$ and ReLu (rectifier linear unit) activation function; c) dense layer (single-layer perceptron) with ReLu activation function and d) output neuron for the prediction (fig. 4).

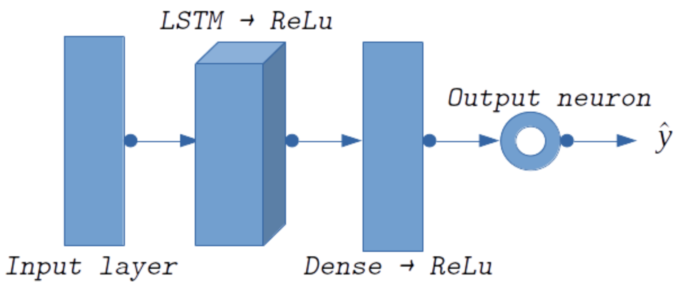

Fig. 4. Architecture of proposed network

The loss function here is the mean squared error defined as following:

$$J^{\{b\}} = \frac{1}{n_b}\sum_{i=1}^{n_b}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 + \frac{\lambda}{2n_b}\sum_r \left\|\omega^{[r]}\right\|_F^2, \tag{1}$$

where $J^{\{b\}}$ is the loss function calculated for each mini-batch, n_b is the number of mini-batches, λ is the regularization parameter, $\hat{y}^{(i)}$ and $y^{(i)}$ are the predicted and actual values, $\omega^{[r]}$ are the weights, $\|\cdot\|_F$ denotes the Frobenius norm, r is the layer's number. To update weights during the training, the adaptive moment estimation (Adam) optimizer was selected.

The process of time series prediction can be split into two approaches: walk-forward validation and multi-step forecasting. To evaluate our model we have chosen the first one, which means a step-by-step forecasting [9]. After the one-step prediction for the test set point, the actual value becomes available to the model.

This numerical schema can be used to predict a new data point, compare it with the incoming value and make a decision about the anomaly of incoming value based on some metrics.

Model hyper-parameters estimation and results. To configure the proposed model and calculate an optimal set of hyper-parameters we selected the grid-search algorithms. Firstly, the multidimensional array of hyper-parameters is generated. We restrict ourselves to the following ones: $num\_nodes = [20, 30, 50, 70, 100]$, $l = [210, 336, 480]$, $num\_epochs = 20$, $num\_mini\_batch = [10, 30, 50, 70, 100]$. Thus, the resulting multidimensional grid $G = [num\_nodes \times l \times num\_epochs \times num\_mini\_batch]$ consists of 75 unique combinations.

For each combination $g_i \in G$ we have performed the training/validation routine using a high-performance GPU Nvidia RTX 2080 Super. The epochs time varied form 3 ms/sample up to 26ms/sample. Having completed the full grid-search we obtained the optimal combinations of hyper-parameters: $num\_epochs = 20$, $l = 210$, $num\_nodes = 50$, $num\_mini\_batch = 20$.

The results of training and prediction made for the calculated optimal hyper-parameters set are depicted in figure 5.

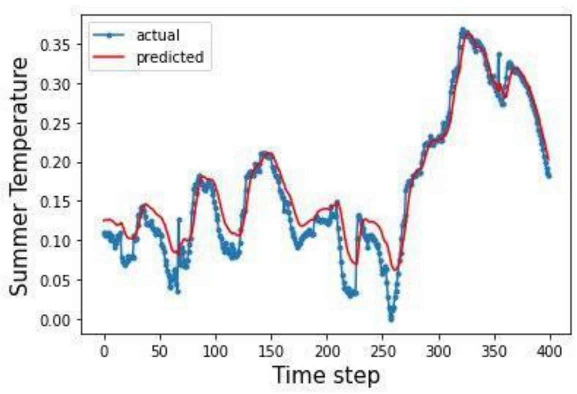

Fig. 5. Prediction results using the walk-forward validation

As it is seen from the provided figures, the model reaches a good performance with the low loss value equals to 6.3710e-04 and root mean squared error equals to 0.025.

STM 32 deployment and X-CUBE-AI environment.

The trained model described in the previous section has been saved in .h5 format in order to be deployed on the microcontroller STM 32. In this section we briefly describe the model integration process.

STM32 is the family of microcontrollers produced by STMicroelectronics. They are widely used in many of IoT applications and have a perspective as a core part of mist computing. X-CUBE-AI is the extension package that allows one to embed and use neural networks on STM32 microcontrollers. This extension provides the possibility to automatically transform pre-trained neural networks with the subsequent integration into the user's project even with limited hardware resources [10].

The created STM32 NN library (both specialized and general parts) can be directly integrated into the IDE project or into the build system via makefile. A well-defined and specific client output API is also exported to develop an AI-based client program. Various frameworks and layer types for deep learning are supported. All basic X-CUBE-AI functions are available through a full and unified command line interface (console level) to perform the basic steps for analyzing, validating, and creating an optimized NN C library for STM32 devices. It also provides post-workout quantization support for the Keras model.

The operational schema of the X-CUBE-AI component is shown in fig. 6.

The project is implemented on a microcontroller STM32F407VGT6, the technical characteristics of which are as follows: 32-bit ARM® Cortex®-M4 with 32-bit FPU core, 1-Mbyte Flash memory, 192-Kbyte RAM, ST-LINK/V2-A on STM32F407G-DISC1, Virtual Com port.

Communication with a PC to download test data and obtain classification results was carried out through the OTG port, where the USB_OTG_FS device had been created. Before using the classifier, the model was analyzed and validated. The complexity in terms of the number of Multiply and Accumulate Operations (MACC) was 2.2 10^6, and 99% of them fell on the LSTM layer. Testing was performed on a time series of length 210 (dimension of the input layer) and obtained a value that corresponds to the output of Keras method model.predict () used in LSTM class implemented in Python.

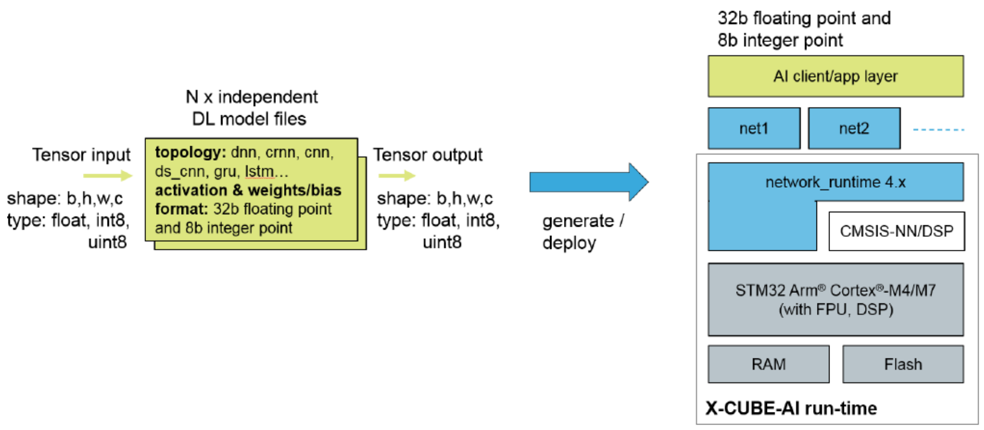

Fig. 6. X-CUBE-AI component

Conclusions and future steps.

In this paper we have described an approach to the development of neuro-controller smart home system. The purpose of such system is to shift out the computational efforts for the temperature data anomaly detection to microcontroller directly connected to the set of sensors. The work is approximately divided into two parts: the model development and training with subsequent deployment on STM 32. Conducted research only deals with the one hidden layer LSTM neural network, whereas the better approaches can utilize ConvLSTM and other hybrid models, which will be a subject of the future studies. The calculated set of hyper-parameters has been obtained using the relatively inefficient grid-search algorithm; hence evolutionary learning will be the more effective method to seek the optimal parameters and neural network architecture. This step is also will be considered and studied in the next research.

REFERENCES

1. *Sabella D.* Multi-Access Edge Computing in Action / D. Sabella, R. Frazao., 2019. – 230 c. – (1 edition).
2. IoT Based Home Automation Using Raspberry Pi / K.Venkatesh, P. Rajkumar, S. Hemaswathi, B. Rajalingam. // Journal of Advanced Research in Dynamical and Control Systems. – 2018. – №10. – C. 1721–1728.
3. *Patchava V. A* Smart Home Automation technique with Raspberry Pi using IoT / V. Patchava, H. Kandala, P. Ravi Babu. // 2015 International Conference on Smart Sensors and Systems (IC-SSS). – 2015. – C. 1–4.

4. *Chakraborty T.* Home automation using edge computing and Internet of Things / T. Chakraborty, K. Soumya. // 2017 IEEE International Symposium on Consumer Electronics (ISCE). – 2017. – C. 47–49.
5. *Batalla J.* Deployment of smart home management system at the edge: mechanisms and protocols / J. Batalla, F. Gonciarz. // Neural Comput & Applications. – 2019. – №31. – C. 1301–1315.
6. Algorithm of Tunning Heating Source Thermophysical Parameters in Smart Home / [O. Sinkevych, L. Monastyrskii, B. Sokolovskii та ін.]. // 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH). – 2020. – C. 9–12.
7. Heating behaviour in English homes: An assessment of indirect calculation methods / T.Kane, S. Firth, T. Hassan, V. Dimitrou. // Energy and Buildings. – 2017. – №148. – C. 89–105.
8. Keras [Electronic source] – Available from: https://keras.io/.
9. *Lopez de Prado M*. Advances in Financial Machine Learning / Marcos Lopez de Prado., 2018. – 400 c. – (ISBN: 978-1-119-48208-6).
10. Getting started with X-CUBE-AI Expansion Package for Artificial Intelligence (AI) [Electronic source]– Available from: https://www.st.com/resource/en/user_manual/dm00570145-getting-started-with-xcubeai-expansion-package-for-artificial-intelligence-ai-stmicroelectronics.pdf

РОЗРОБКА НЕЙРО-КОНТРОЛЕРА НА БАЗІ STM 32

О. Сінькевич, Л. Монастирський, Я. Бойко, Б. Соколовський

*Львівський національний університет імені Івана Франка,
вул. Драгоманова, 50, 79005, Львів, Україна*
oleh.sinkevych@lnu.edu.ua

У цій роботі розглянуто покроковий підхід до розробки системи нейро-контролера. Для використання мікроконтролера STM 32 в якості апаратного рішення для граничних обчислень обчислень ми пропонуємо розгортання моделі глибокого навчання для задач прогнозування та виявлення аномалії. Дані, які використовуються як для тренування моделі, так і для тестування, складаються з часового ряду кімнатної температури розумного будинку з наявними часовими залежностями. Розроблена модель глибокого навчання на базі архітектури LSTM автоматично виявляє залежності від часу під час тренування та використовує їх для безпосереднього тестування. Для її розробки був застосований високорівневий фреймворк Keras, який дозволяє порівняно швидко створювати робочі прототипи, що згодом можуть бути конвертовані у формат, який підтримується TensorFlow Lite з метою їх подальшого розгортання на граничних пристроях та мікроконтролерах з підримкою глибокого навчання. Наведений етап підготовки даних для задачі прогнозування з подальшим перевизначенням задачі до проблеми навчання з вчителем, що передбачає перетворення вхідних нормалізованих даних з використанням параметру, який визначає довжину предикторів деякого поточного значення часового ряду. Для обчислення набору оптимальних гіпер-параметрів LSTM застосовано алгоритм пошуку на сітці можливих значень, що передбачає поетапний процес повного навчання нейронної мережі. Тренування розробленої нейронної мережі було здійснено на графічній платі Nvidia RTX 2080 Super, що дозволило відносно швидко здійснити перебір заданих параметрів. Навчена мережа була розгорнута та протестована на мікроконтролері STM 32 із встановленим пакетним розширенням X-CUBE-AI. Отримані результати валідації та тестування показують можливість використання такого підходу для розробки та вдосконалення архітектури LSTM для граничних та туманних обчислень, що буде проведено у наступних дослідженнях.

Ключові слова: граничні обчислення, глибоке навчання, LSTM, розумний будинок, STM 32

Стаття: надійшла до редакції 22.05.2020,
доопрацьована 01.06.2020,
прийнята до друку 03.06.2020.