Using machine learning for cognitive Robotic Process Automation (RPA)

Pedro Martins, Filipe Sá, Francisco Morgado, Carlos Cunha

Polytechnic Institute of Viseu, Informatics Department, Viseu, Portugal
pedromom@estgv.ipv.pt, filipe.sa@estgv.ipv.pt,
fmorgado@estgv.ipv.pt, cacunha@estgv.ipv.pt

***Abstract*—There are many business routine tasks and processes which are performed by qualified resources which can be reallocated, allowing qualified workers to dedicate their effort to other fields/tasks with more relevance. With Robotic Process Automation (RPA), repetitive tasks, even complex computer routine processes, can be automated and improved, independently of the interface or menu location.**

This paper proposes an RPA application, which in real-time, dynamically detects objects in software applications interface, allowing flexibility, performance, and accuracy regardless of the OS, location of interface tools, and necessary menus to reach it. For this purpose, a Convolution Neural Network (CNN) is trained with several interfaces and menus and used to classify software interfaces in real-time. Furthermore, a developed software takes automated actions, moving the mouse pointer, clicking, editing text, and performing any necessary action.

Results show that the proposed RPA technique based on deep learning is capable of detecting objects in real-time, classify them, with outstanding accuracy, and take dynamic actions. This way, it is possible to automate any computer task.

***Keywords*-Machine learning; Convolution neural network (CNN); TensorFlow; YOLO; Cognitive; Robotic Process Automation (RPA);**

I. Introduction

Robotic Process Automation (RPA) is a technology which works with machine learning to detect objects on the screen and take actions. Typically when a programmer wants to automatize a list of tasks for a given objective as a work-flow, APIs and dedicated scripts must be programmed are used. With RPA, the automation process mimics the programmer actions by observing and recording the manual execution using the Graphical User Interface (GUI). The RPA stores each menu click and key, then the process is repeated as many times as necessary, independently of the location of menus on the screen. Comparing with traditional approaches, where recording the location of each click and the key is used for later replication. If menus change place, shape, screen resolution, operating system, or even if some error occurs, fixed recorded scrips will no longer be valid or function as desired.

In this work is proposed the use of deep learning neural networks to identify objects on-screen interfaces, to accomplish it, the neural network is trained, and used to detect objects and then autonomously actuate with configured actions. This learning and repeating process allow even the automation of programs that do not have support for APIs [6]. TensorFlow is a framework oriented to deep learning, simplifying the implementation of CNN's oriented to object detection [1]. The tool CNN YOLO (You Only Look Once) is used inside TensorFlow to detect objects in real-time and extract features which allow to decision making and take action. Since object detection and reaction needs to be in real-time, YOLO algorithm shows to have the most promising characteristics regarding training and classification time [12].

Based on our test, YOLO CNN shows good and promising classification results. After objects detection, another app handles the coordination of actions based on the specific location of the objects, moving the mouse pointer to click or perform any other configured action.

The organization of this document is as follows. Section II, describes the motivation for this work and the most relevant contributions. Section III, resumes the vast related-work in the field. Section IV introduces the preliminary steps to reach results. Section V shows some of the obtained results. Finally, Section VI concludes the work with a critical review and proposes future work guidelines.

II. Motivation and contributions

This work takes place in the Polytechnic Institute of Viseu, Department of Computer Sciences, providing support to the IT department, for which the proposed tool automates many repetitive processes, some of them of complex execution. For instance, at the detection that a particular server is down, an automated script takes action to restart the server back online. Other tasks include doing routine maintenances such as configuring operating systems and installing applications. Another relevant application of RPA is for students with special needs (e.g., vision problems, and concentration problems, which difficult the usage of the mouse and keyboard with precision). The proposed cognitive RPA aims to help students performing tasks, many of them in the field of the computer science course, such as networks configuration. The contributions present in this paper are:

- A new paradigm on task scripting;
- Labelled dataset on software interfaces;
- A trained CNN's on interfaces;
- RPA integration with deep learning;
- Result analysis of object training and classification;

2020 15th Iberian Conference on Information Systems and Technologies (CISTI)
24 – 27 June 2020, Seville, Spain
ISBN: 978-989-54659-0-3

- Practical application of deep learning and classification in real-time techniques, for RPA scripting/action.
- A tool for business IT and also applicable to people with cognitive problems.

III. Related work

Currently, there are several frameworks oriented to deep learning [17]. The available frameworks allow an abstraction level which makes easy the creation of new training models for several neural networks API. Most popular frameworks include:

- Google initially development was Tensorflow; this framework is a leader, providing API's for several programming languages (e.g., Python, C++, Haskell). The architecture includes three primary levels of operation - first, a low level based in C++ and CUDA. Second, an API in Python oriented to graphs. Third, a high-level API, such as TF-learn, TF-slim, or Keras [4]. All of these features meet the desired requirements for this research work.
- PyTorch, a widely used framework on Facebook, which includes a set of wrappers and support for other languages. Two high-level features are provided, Computation based on GPU (tensor) such as "NumPy", and support for deep neural networks [11] [2].
- Caffe2 (Convolution Architecture for Fast Feature Embedding), is a Python-based framework for deep learning, specially developed to express any model, like CNN, RCNN, LSTM, and fully connected neural network designs, based on a friendly API developed in Python with support for C++ and CUDA [9] [3].
- Microsoft Cognitive Toolkit, harness the intelligence within massive datasets through deep learning by providing uncompromised scaling, speed, and accuracy. This toolkit describes neural networks as a set of steps through an oriented graph, where leaf knots represent input parameters, while the remaining represent operations over those inputs [5] [10].
- MatLab, is a popular tool and IDE for deep learning models programming, without being an expert in the field. MatLab includes several tools and models including, GoogLeNet, VGG-16, VGG-19, AlexNet, ResNet-50, ResNet-101, and Inception-v3. At the same time, graphical acceleration based on CUDA is already integrated, as well as ground-truth labelling for image and video. MatLab can also work with models from Caffe and TensorFlow-Keras and allows to collaborate with peers using frameworks like PyTorch and MxNet [13] [15].

In other fields, researchers are promoting RPA in the financial area [14] in the context of trading, affecting accounting staff involved in the banking area [16]. Artificial Intelligence (AI) is a solution for big data and provides a new possibility for accurate prediction of RPA. Sample vendors include Advanced Systems Concepts, Automation Anywhere, Blue Prism, UiPath, and WorkFusion [8].

The proposed work stands out from others in the field of RPA and deep learning techniques integration, applied for real-time classification and script action. In this context, RPA'a also benefits from deep learning and other artificial intelligence algorithms, for instance, for a given task, detect that a specific app is required, or impose security measures by preventing particular actions. Thus, in this case, the RPA application can autonomously install new software and guarantee security.

In the field of RPA research domain, for the best of our knowledge, little has been done in cognitive computer tasks automation and integration involving deep-learning networks. In [7], authors use RPA to automate human action and support efficient business processes. A software robot is used instead of humans for typing and clicking and even analyzing data in different applications. However, there is no integration of neural networks to detect where to click or where to write the text. If elements change of place on the screen, the script will no longer produce a valid result.

IV. Experimental setup

For all implementation and experiments the following hardware is used: CPU i7 3.6GHz; 16GB RAM; 3TB HDD; Asus Nvidia GeForce GTX 1070 8GB.

A dataset is necessary to train the neural network. As far as our search went, there is no dataset on software interfaces. Thus, the dataset was created and labelled by hand during the development of this research work. Additional datasets are used to perform the CNN validation and testing. The dataset contains 10.000 snapshots of different applications software interfaces (all available for Windows 10), such as full interface snapshots; configuration panels; tools; option and configuration menus; and institutional web pages. These snapshots have different resolutions, themes, colours, and orientations. Additionally, the dataset contains three main groups, training (60%), testing (20%) and validation (20%).

Labelling is done manually, with the help of our research group. This task is done image by image, using a graphical image annotation tool, LabelImg [18]. Figure 1 shows an example of label mapping for buttons and menus, where the red bounding boxes map the coordinates of each button and menu to a given label. The output of this monotonous task is n XML, containing coordinates and labels with names.

All annotations are in XML using the PASCAL VOC format. Next step, they are converted to a CSV format, with a custom script that generates a file with all label mappings ('labelmap.pbtxt') which is used to train the YOLO v3 algorithm.

For the sake of simplicity, we focus only on the screenshots of one application, Eclipse IDE. After training YOLO with 16.000 iterations, to detect buttons and menus, another developed script takes action, moving the pointer and clicking. Note that, any action can be set to be performed, in this work is only presented a simple example as proof of concept.

V. Evaluation analysis

During the evaluation tests, the Eclipse interface was explored and labelled using different colour themes and sizes. After the labelling process, the training process of TensorFlow

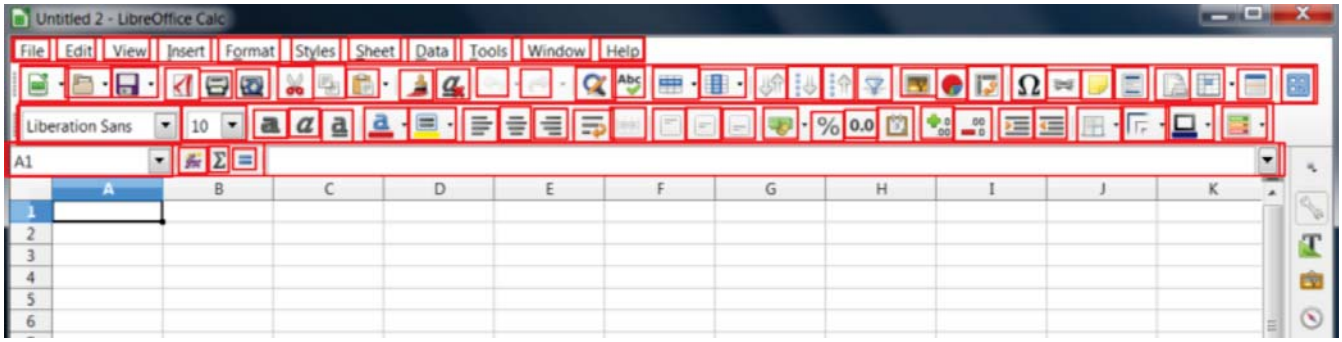

Fig. 1: Label mapping example

TABLE I: Checkpoint 1 - average precision values

	True Positives	False Positives
Help	53%	64%

TABLE II: Checkpoint 2 - average precision values

	True Positives	False Positives
File	82%	71%
Edit	73%	-
Navigate	99%	-
Search	53%	-
Project	99%	-
Form	92%	-
Run	95%	-
Window	98%	-
Help	97%	63%

with YOLO started. As a checkpoint, to verify if the training process was running as desired, after 891 iterations, the training process was stopped, and the classification and location loss values recorded.

Figure 2, shows the classification loss, and the location loss. The classification loss (on the left) shows the correctness of the classification of the objects, the lower this value is, the better. The location loss (on the right), shows the error between the detected bounding boxes and the ground truth (the lower this value becomes better).

Figure 3 shows the first attempt to classify the "Help" button. However, because the training process is at an early phase, there are many false positives detected, as well as wrong locations. In 3 and Table I, is possible to see that the "Help" menu was wrongly classified and at the same time the bounding boxes are not in the correct location. Nevertheless, with 53% certainty, the "Help" menu is correctly classified.

Next checkpoint at 3534 iterations demonstrates a much higher classification success, Table II. Although precision is still low, and there are some false positives. Figure 4, depicts the result of the classification at checkpoint 2, where several false positives are present. Comparing the false positives values from Table I and Table II, it is possible to see them decreasing, simultaneously the true positives are increasing.

TABLE III: Checkpoint 3 - average precision values

	True Positives	False Positives
File	89%	51%
Edit	99%	-
Navigate	99%	-
Search	96%	-
Project	99%	-
Form	98%	-
Run	80%	-
Window	99%	-
Help	99%	-

Figure 5 and Table III, shows the last checkpoint at 16495 iterations. Most of the tested classes are correctly classified, with precision around 99%, except for the "File" button class, which overall tests demonstrated the lowest precision with overlapping of the bound box location. Note that, the false-positive on the "Help" menu disappeared and global precision improved substantially.

Figure 6, shows the Average Precision (AP) only for the "File" and "Help", which are two extremes, one is well detected, and the other demonstrated many false positives during the entire process. In the chart from the left side, the "File" menu average precision struggles to improve. In the right side, the "Help" menu is near the 90% precision.

VI. Conclusions and Future Work

For this work, three main types of distinct tests are performed using the TensorFlow framework and YOLO v3 deep learning network.

- The first group of tests (focused on this paper), comprehends the detection of different components on different versions of programming IDEs. The average detection precision (mAP) for all tested samples was 95.6%.

- Second group of tests - different colours and components on screens (for which the model did not receive training). Results for the average precision variate between 70% and 93.32%.

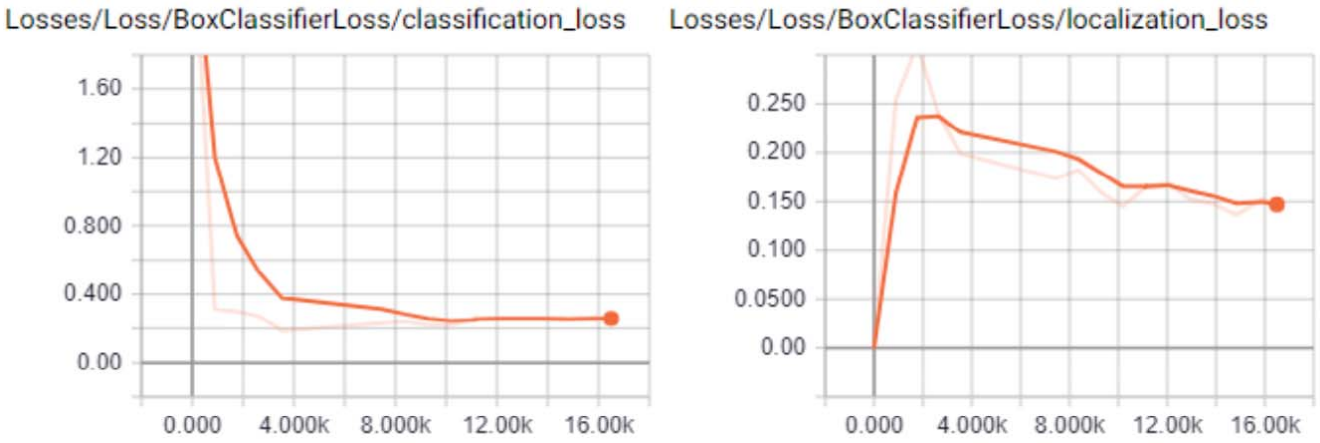

Fig. 2: Checkpoint 1 - Losses for the Final Classifier

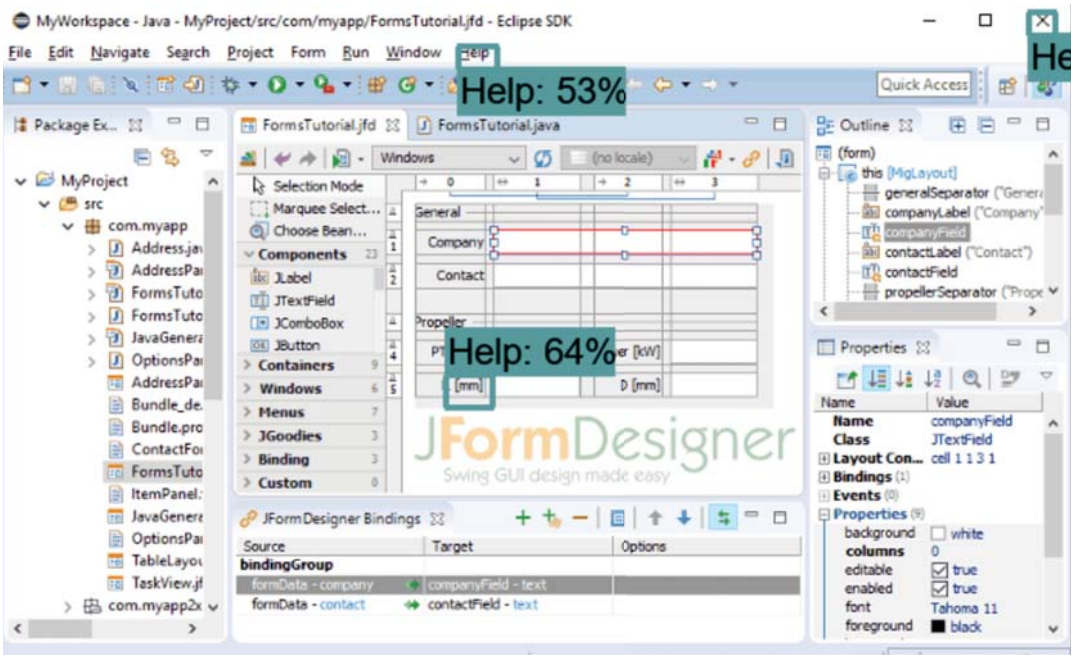

Fig. 3: Checkpoint 1 - Eclipse “Help” button, false positives

- In the final group of tests, changes to the images sizes and resolutions are made. Components were re-sized in and out, losing quality.

Nevertheless, with before mentioned modifications, the results keep the original mAP. Moreover, given that the CNN is training with a dataset that suffers from low variations when compared with typical photos, the absence of many factors (e.g. light, angle, shadows) influence the accuracy of the network, leading it to approximate perfection (almost 100%) as the dataset size and completeness increases. At this point, the only technical limitations found, lie on the size of the dataset, which requires more work, adding more screens and labelling. On the practical side, the tool has difficulties in acting like a human when performing login and registration actions. The human validation problem requires training in the identification, interpretation and actuation on the captcha and other mechanisms.

In general, this study shows the enormous potential of YOLO for object detection and applications-oriented to RPA.

Future work focuses on actuation; this is, after detecting the objects, independently actuate with the most appropriated action. Other research directions consist of endow the RPA with more in-depth intelligence; for instance, autonomously open and install apps when required for a given task. As a direct consequence of our future work development, a larger and more complete dataset will be built and made available.

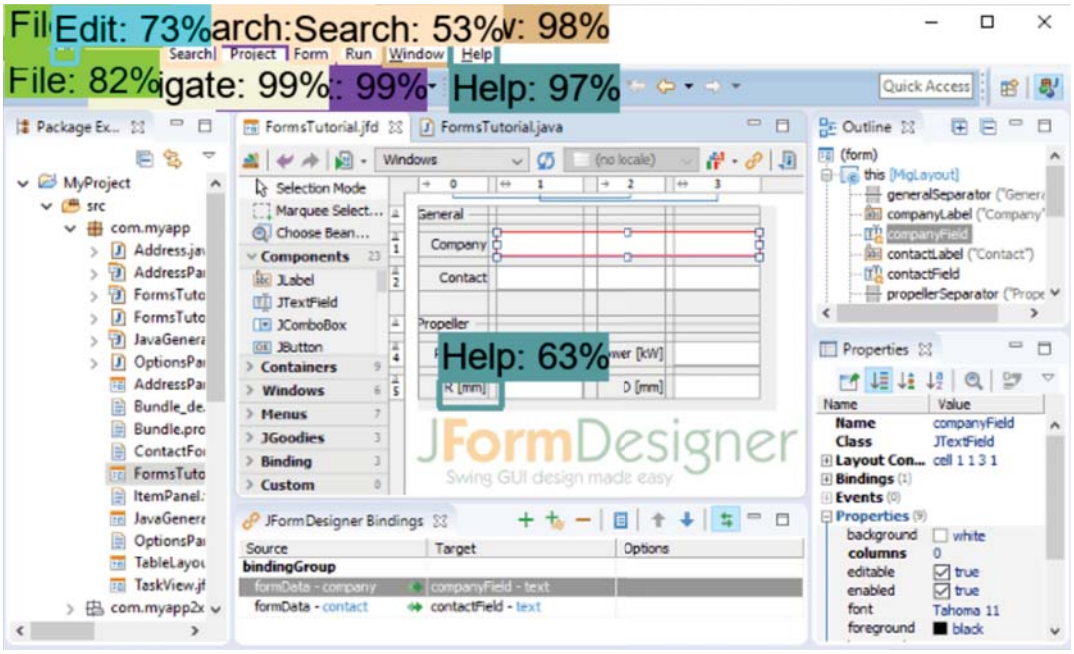

Fig. 4: Checkpoint 2

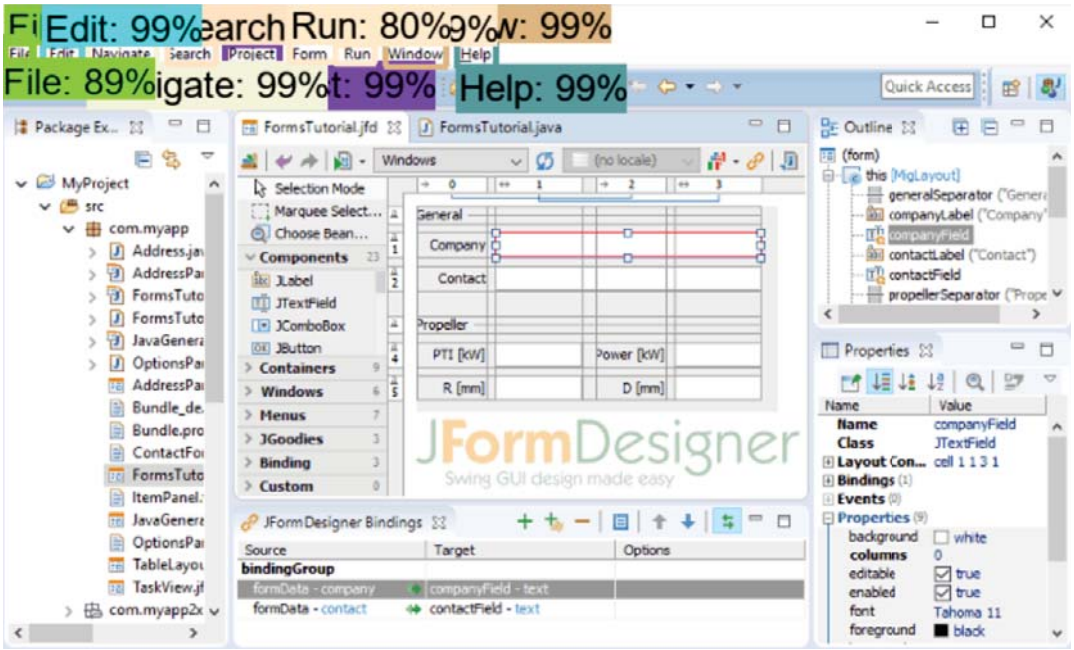

Fig. 5: Checkpoint 3 - final

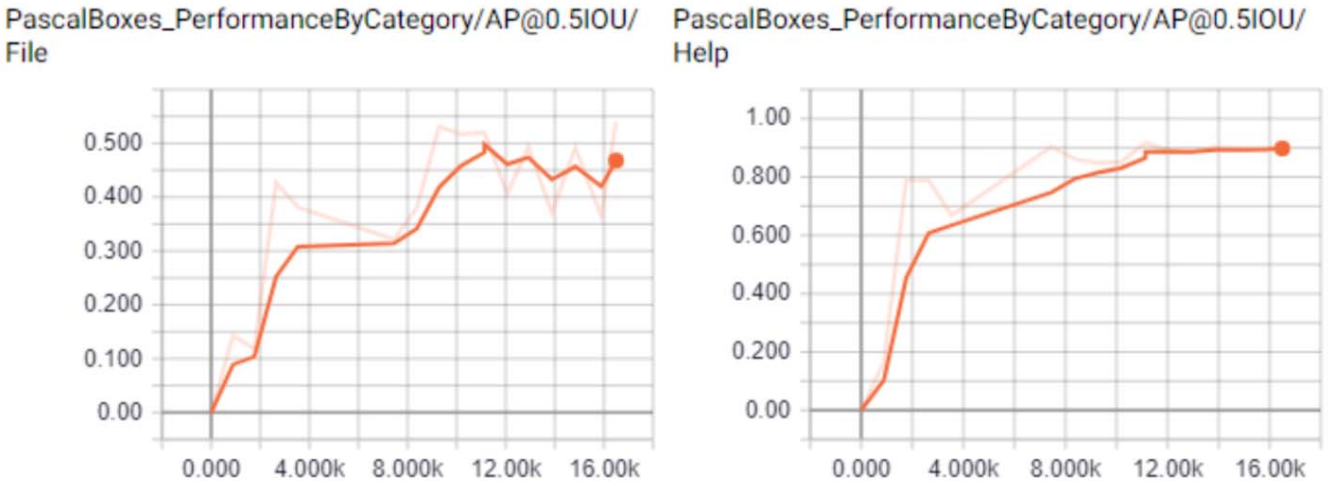

Fig. 6: Comparing the Average Precision (AP) for the "File" and "Help"

ACKNOWLEDGEMENTS

"This work is funded by National Funds through the FCT - Foundation for Science and Technology, I.P., within the scope of the project Refa UIDB/05583/2020. Furthermore, we would like to thank the Research Centre in Digital Services (CISeD), the Polytechnic of Viseu for their support, and Softinsa IBM subsidiary."

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] F. Chollet. *Deep learning with python*. Manning Publications Co., 2017.

[3] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *High Performance Computer Architecture (HPCA), 2018 IEEE International Symposium on*, pages 620–629. IEEE, 2018.

[4] A. Heinecke, E. Georganas, K. Banerjee, D. Kalamkar, N. Sundaram, A. Venkat, G. Henry, and H. Pabst. Understanding the performance of small convolution operations for cnn on intel architecture. 2017.

[5] K. Kharchenko, O. Beznosyk, and V. Romanov. Implementation of neural networks with help of a data flow virtual machine. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pages 407–410. IEEE, 2018.

[6] M. Lacity, L. P. Willcocks, and A. Craig. Robotic process automation at telefonica o2. 2015.

[7] M. C. Lacity and L. P. Willcocks. A new approach to automating services. *MIT Sloan Management Review*, 2017.

[8] H. Lu, Y. Li, M. Chen, H. Kim, and S. Serikawa. Brain intelligence: go beyond artificial intelligence. *Mobile Networks and Applications*, 23(2):368–375, 2018.

[9] D. Novak, M. Batko, and P. Zezula. Large-scale image retrieval using neural net descriptors. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 1039–1040. ACM, 2015.

[10] D. Parra and C. Camargo. A systematic literature review of hardware neural networks. In *2018 IEEE 1st Colombian Conference on Applications in Computational Intelligence (ColCACI)*, pages 1–6. IEEE, 2018.

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[13] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. 2018.

[14] D. Tran, T. Ho Tran Minh, et al. Workflow methodology development of rpa solution for a vietnamese bank: A case study of korkia oy. 2018.

[15] A. Vedaldi, M. Lux, and M. Bertini. Matconvnet: Cnns are also for matlab users. *ACM SIGMultimedia Records*, 10(1):9, 2018.

[16] S. Vishnu, V. Agochiya, R. Palkar, et al. Data-centered dependencies and opportunities for robotics process automation in banking. *Journal of Financial Transformation*, 45:68–76, 2017.

[17] G. P. Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.

[18] X. Zhang, L. Duan, L. Ma, and J. Wu. Text extraction for historical tibetan document images based on connected component analysis and corner point detection. In *CCF Chinese Conference on Computer Vision*, pages 545–555. Springer, 2017.