A Context-Aware Machine Learning-based Approach

Conference Paper · October 2018

CITATIONS
10

READS
3,456

4 authors:

Nathalia Nascimento
University of Waterloo
28 PUBLICATIONS **142** CITATIONS
SEE PROFILE

Paulo Alencar
University of Waterloo
162 PUBLICATIONS **1,650** CITATIONS
SEE PROFILE

Carlos Lucena
Pontifícia Universidade Católica do Rio de Janeiro
713 PUBLICATIONS **7,768** CITATIONS
SEE PROFILE

Donald D Cowan
University of Waterloo
317 PUBLICATIONS **3,202** CITATIONS
SEE PROFILE

A Context-Aware Machine Learning-based Approach

Nathalia Nascimento*
Laboratory of Software Engineering
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
nnascimento@inf.puc-rio.br

Paulo Alencar
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
palencar@csg.uwaterloo.ca

Carlos Lucena
Laboratory of Software Engineering
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
lucena@inf.puc-rio.br

Donald Cowan
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
dcowan@csg.uwaterloo.ca

ABSTRACT

It is known that training a general and versatile Machine Learning (ML)-based model is more cost-effective than training several specialized ML-models for different operating contexts. However, as the volume of training information grows, the higher the probability of producing biased results. Learning bias is a critical problem for many applications, such as those related to healthcare scenarios, environmental monitoring and air traffic control. In this paper, we compare the use of a general model that was trained using all contexts against a system that is composed of a set of specialized models that was trained for each particular operating context. For this purpose, we propose a local learning approach based on context-awareness, which involves: (i) anticipating, analyzing and representing context changes; (ii) training and finding machine learning models to maximize a given scoring function for each operating context; (iii) storing trained ML-based models and associating them with corresponding operating contexts; and (iv) deploying a system that is able to select the best-fit ML-based model at runtime based on the context. To illustrate our proposed approach, we reproduce two experiments: one that uses a neural network regression-based model to perform predictions and another one that uses an evolutionary neural network-based approach to make decisions. For each application, we compare the results of the general model, which was trained based on all contexts, against the results of our proposed approach. We show that our context-aware approach can improve results by alleviating bias with different ML tasks.

*CAPES scholarship/Program 194/Process: 88881.134630/2016-01

CASCON'2018, October 2018, Markham, Ontario Canada

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Learning settings*; **Intelligent agents**; • **Applied computing** → **Computers in other domains**; • **Software and its engineering** → **Layered systems**; **Software system models**; **Abstraction, modeling and modularity**; **Software design engineering**;

KEYWORDS

context-awareness, learning bias, machine learning, neural network, contextual modeling

ACM Reference Format:
Nathalia Nascimento, Paulo Alencar, Carlos Lucena, and Donald Cowan. 2018. A Context-Aware Machine Learning-based Approach. In *Proceedings of Annual International Conference on Computer Science and Software Engineering (CASCON'2018).* CASCON, Markham, ON, Canada, 8 pages.

1 INTRODUCTION

In the last decade, researchers have worked on the problem of how to create general and versatile machine learning (ML)-based models to enable systems to handle complex tasks. For example, a classification task that uses an ML model to process a dataset that contains data from monitoring environments [7], or a control task that uses an ML model to assist a set of robotic agents in analyzing and interacting with a dynamic environment [14]. These kinds of tasks usually involve a great number of contexts, that is, information that can be used to characterize the state of different entities, such as persons, places or physical objects [1].

A versatile model must work well on all operating contexts. However, according to [12], "as the volume of data increases, the learner may become too closely biased to the training set and may be unable to generalize adequately for new data." Consequently, this model tends to generate disproportionate weight in favor of or against a specific context. Even when the bias is not so evident, it can represent a significant impact for critical applications, such as environmental monitoring, energy consumption control [10], healthcare scenarios or air traffic control, which are the kinds of applications that must produce decisions that are timely, robust and secure.

Learning bias is not a new problem. In 1992, Bottou and Vapnik [2] proposed the strategy of local learning to alleviate this problem. The local learning strategy consists of separating the input space into subsets and then building a separate model for each subset. Bottou and Vapnik [2] state that "such an algorithm looks both slow and stupid...Empirical evidence however defeats this analysis. With proper settings, this simple algorithm improves significantly the performance of our best optical character recognition networks." Although local learning itself is not a novel concept, it has recently gained credibility because of the natural complexity of current applications [10], specifically those that must deal with large datasets. Notwithstanding, Stone [16] proposed the strategy of task decomposition, which is a popular approach for learning complex control tasks in robotic systems [17]. This strategy consists in breaking complex tasks down into small tasks, and then using a given algorithm (or a set of algorithms) to operate on each task. For example, in a simulated soccer game where a team of agents uses a neural network to make decisions, Whiteson et al. [17] propose to break this complex neural network into four simple networks based on the tasks that each agent performs during the game.

In this paper, we propose an approach that is suitable for both classification and control tasks. Our method consists of a local learning-based approach that uses contextual information to separate the input space, train and select the machine learning models. Context-aware computing is used to represent a system that understands the context and takes an action based on that particular context [15]. Accordingly, our context-aware machine learning-based approach involves: (i) anticipating, analyzing and representing context changes; (ii) training and finding machine learning models to maximize a given scoring function for each operating context; (iii) storing trained ML-based models and associating them with corresponding operating contexts; and (iv) deploying a system that is able to select the best-fit ML-based model at runtime according to the context.

There are some application scenarios that we believe that could be improved by our context-aware approach. An example involves a smart solar battery that is used in aircraft, as suggested in [9], which controls energy consumption in order to maximize the flight time. Since the sunrise time varies in many countries over the year, we pose the following question: will the results be better if we create a general model to control this solar battery, independent of the season, or if we create a solution that considers a temporal context and is able to switch to specialist models based on the season?

To evaluate the proposed approach, we present two experiments in Section III. This section presents the experimental setup, results, and evaluation. The remainder of this paper is organized as follows. Section II describes the proposed approach. Section IV presents related work. The paper ends with concluding remarks in Section V.

2 APPROACH

The first step of our proposed approach is to identify the attributes we need to consider in contextualizing the information and to determine the contexts that this application will consider. For example, if we have an application that contains a set of robots that interact with a place, this place can be characterized by the temperature and the background lighting information. Thus, the different contexts for this application will only vary the temperature and background lighting information. If we have an application that uses a glove for hand-gesture recognition, we may consider the hand shape information as a contextual information in evaluating different models. In an application for self-tracking, which monitors a person's data and performs disease prediction, a person can be distinguished based on the country of residence. The information that we select to characterize one or more entities in the application will be used as a parameter to detect changes, to separate the input space, to train and to deploy the ML models.

Our architecture consists of two modules, as depicted in Figure 1: (i) a system that uses a machine learning model to analyze a dataset or interact with an environment while a learning algorithm tries to maximize a given scoring function; and (ii) a control module that configures machine learning models, monitors the context changes, and maintains a history of the trained ML-based models that provide the best result for each one of the operating contexts. When the context changes, the quality of the results achieved by the ML model that is in current use may decrease. As a consequence, if context changes are known, the controller will revert to a previous configuration instead of retraining itself. Otherwise, it will test new machine learning models. To allow the system to find a new model, our architecture supports manual and automatic controllers to manage permissible run-time adaptations caused by environmental or system changes. We provide more details about the control module in Section 3, in which we provide detailed explanations about the experiments.

In short, to illustrate the use of our approach, consider a smart solar battery in an aircraft that uses a trained artificial neural network (ANN) to make decisions. In this illustrative example, by using a temporal context, we can evaluate if the results can be improved if the model that we use in the winter is different from the one in the summer. For example, suppose that during the training step, we observed that during the summer, a neural network with binary activation function outperformed a neural network with sigmoid function, but during the winter, the ANN with sigmoid was better. Then, based on our proposed architecture, the solar battery will be deployed with two models: one for the summer and another for the winter. It is an important approach for this kind of application, where accuracy is the most important factor.

3 APPLICATION SCENARIOS

To give more details about and illustrate the use of our proposed approach, we selected two papers in the machine learning literature and reproduced their experiments using our proposed architecture. The first experiment is a prediction task, that consists of using a dataset to train an ML model for air quality prediction, as presented in [7]. The second experiment is a control task, that consists of using a simulated scenario to train an ML model in order to set the behavior of a group of autonomous street lights that must achieve some non-functional requirements, such as performance, as presented in [8].

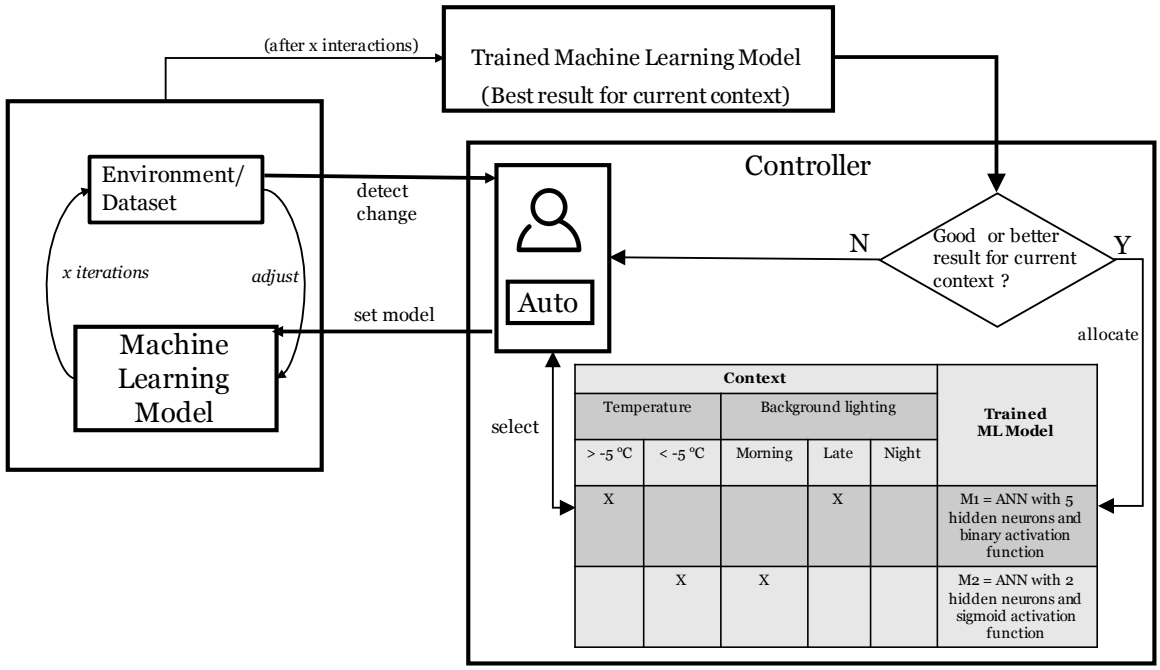

Figure 1: Schematic illustration of our architecture to store and retrieve machine learning models based on context.

3.1 Prediction Task

De Vito et al. (2008) [7] monitored the air quality from a main road located in an Italian city during a period of 13 months. For each measurement, the authors registered the temperature, humidity and some gas sensor readings, such as CO and benzene concentrations. In [7], the goal of the authors is to calibrate the sensor to measure benzene gas, by predicting the benzene concentration based on the other environmental measurements. To predict the benzene concentration, they proposed and implemented a regression schema based on a neural network architecture. They evaluated their approach based on the estimation error of the predictions. They divided and allocated the dataset into three parts: a segment of 8 months randomly selected (≈ 62%) to a training set, ≈ 14% to a validation set, and ≈ 24% to a testing set. Accordingly, they achieved a mean absolute error (MAE) of 0.0337 and a squared correlation coefficient (SCC) - that is a metric that represents the predictive power of the model as a value between 0 and 1 - of 0.9998.

3.1.1 Problem. To make predictions, the authors implemented a neural network with five hidden neurons that was trained by considering the full range of all of these variables. For example, their model was generic enough to predict the benzene concentration independent of the time and of the temperature and humidity of the environment. However, the authors observed that the mean absolute error of the predictions provided by the generic model was higher during the winter time, specially in November. Accordingly, the authors explained that the concentration and dynamics of gases from October to March (coldest months in Italy) are different from April to September, because of the low temperature.

3.1.2 Context-aware-based prediction. As described, De Vito et al. (2008) [7] observed that the prediction error of their generic model varied according to seasonal differences. Thus, we decided to investigate if we can achieve better results by providing different learning models for different seasons.

In this case, we contextualize the air quality data based on the month. We split the dataset into two groups: (i) "April-September" and (ii) October-March. For each group, we allocated 70% to the training set, 10% to the validation set and 20% to the test set. Then, we trained three models: (i) one considering only data collected from April to September; (ii) one considering only data collected from October to March; and (iii) one considering all data.

Figure 2 depicts the training process used in this experiment. As shown, we first trained a generic ML model, not differentiating between the seasons. Then, for each monthly range, we trained specialized models by using specific parts of the dataset. For instance, the three models are based on a Neural Network Regression module with twenty hidden neurons, and they differ only in their weight values. However, as we are proposing a dynamic model, we could have provided different machine learning approaches based on context changes, such as the use of a linear regression-based approach. We used only Neural Network Regression-based models because this kind of approach usually presents better results for this kind of application.

After training the three models, we verified which model provides the lowest absolute error mean for each range of months. As shown in Figure 3, we consider the November month as a specific context, as November is the worst month for the general model presented by De Vito et al. (2008) [7]. Therefore, we tested which

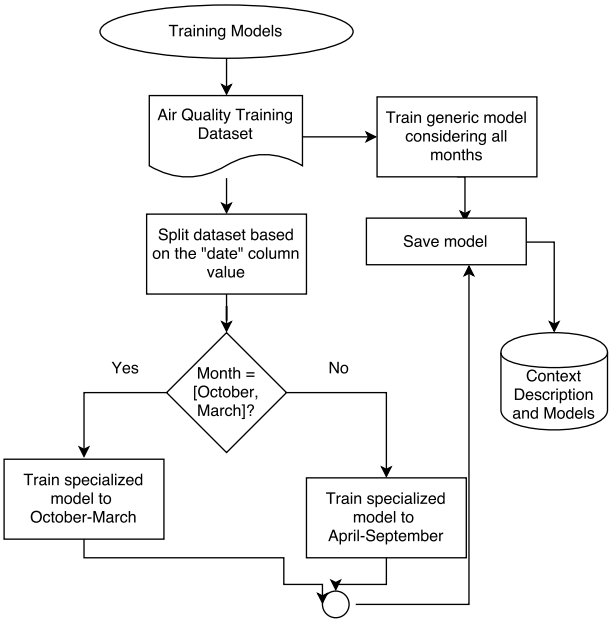

Figure 2: Training step: splitting the dataset based on the month and generating generic and specialized ML models.

model presents the best measurement results between April and September, between October and March, and in November.

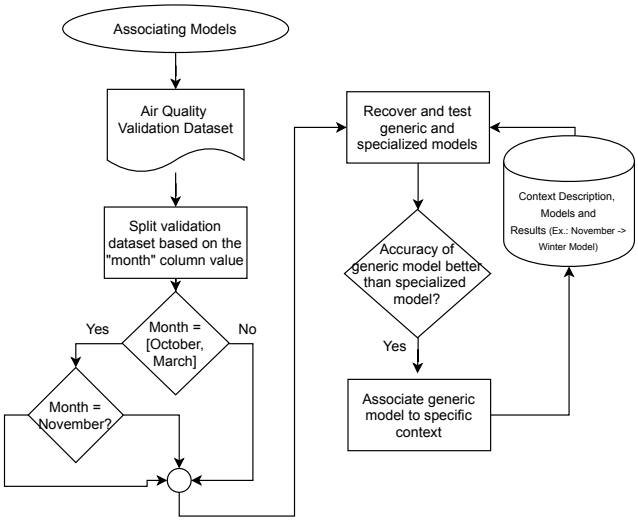

Figure 3: Validation step: associating models to contexts based on validation results.

The model that presents the best results for each monthly context is stored as the model that will be used in that context. For example, if the winter model presents a better result for the November measurements than the generic model, the winter model will be allocated to the November month context. Then, during November, the system will use the winter model to make predictions, as we illustrate in Figure 4.

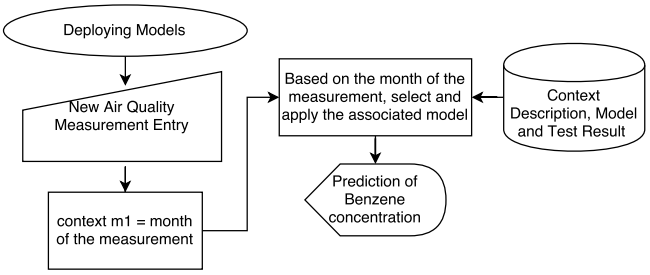

Figure 4: Deployment step: using the highest month score model to provide predictions for the user.

3.1.3 Results. As we explained, we used the same structure of the neural network architecture in all models. In such a case, they only varied in accordance with the training set. We performed this experiment many times, increasing the number of learning iterations (i.e. the maximum number of times the algorithm processes the training cases).

Considering that De Vito et al. (2008) [7] present a mean absolute error (MAE) of 0.0337, our generic model achieved a value higher than this MAE after 500 iterations, as shown in Figure 5. The best result achieved by our generic model was after 30,000 learning iterations, presenting an MAE of 0.025414 and a squared correlation coefficient (SCC) of 0.999984. To provide the following comparisons with the specific models, we selected the generic model that presented the best result for the full validation set.

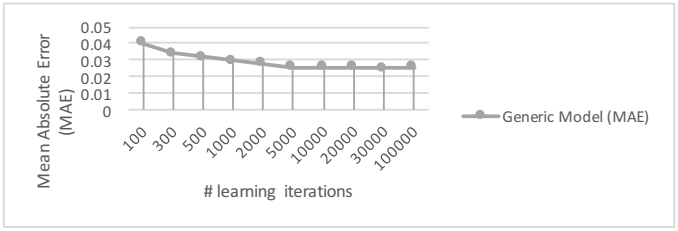

Figure 5: Mean absolute error (MAE) achieved by the generic model considering the full validation set.

Figures 6 and 7 compare the use of the specific models against the generic model based on the context. We observed that from October to March, the generic model presents results better than the specific winter model, independent of the number of learning iterations.

For data collected between April and September, the specific model is considerably better than the generic model if the number of learning iterations is less than 2000. After only 2000 interactions, the generic model started to present a result better than the specific model. In such a case, if the time of training/retraining is the most important factor of the solution, and the system must limit the number of learning iterations to under than 2000, the system will provide better results during the summer if it uses the summer model.

However, considering that the system does not have limited time and the training step can perform more than 2,000 iterations, the

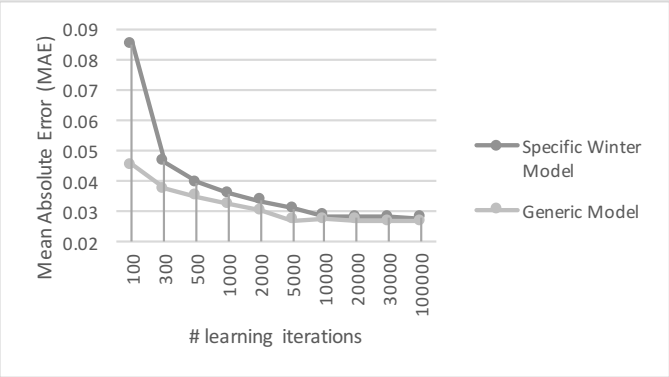

Figure 6: Mean absolute error (MAE) achieved by the generic and winter models for the "October-March" validation set.

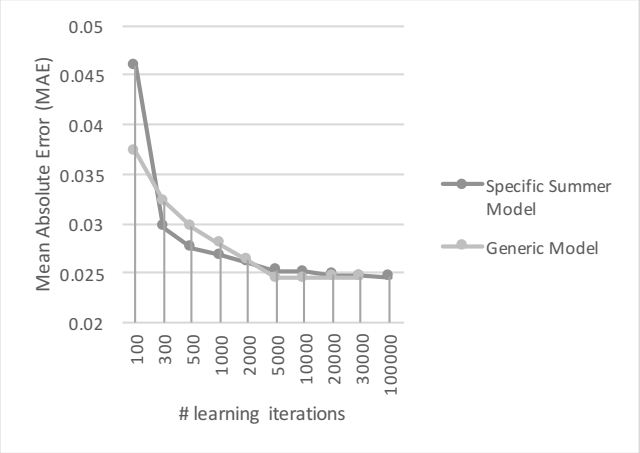

Figure 7: Mean absolute error (MAE) achieved by the generic and summer models for the "April-September" validation set.

generic model will provide a better result during the summer and winter months. The only exception occurs in November, where the best winter model provided a MAE of 0.026048 and the best generic model provided a worse MAE of 0.026204. Therefore, we conclude that the best system configuration is that one that uses the generic model during the whole year, but switches to the winter model in November.

After the validation model, we tested the models using the testing set and the results were in accordance with the results presented during the validation step. For example, the MAE achieved by the generic model for the winter test set was 0.02586, while the MAE achieved by the specific model was 0.026123.

We used the Microsoft Azure ML Studio [5] for these experiments.

3.2 Decision Task

Nascimento and Lucena (2017) [8] described a set of street lights, which are distributed in a simulated neighborhood, with neural networks in order to enable them to make decisions based on the data collected from the environment. They used a genetic algorithm to train the neural networks. For more details concerning this application scenario and the training algorithm, see [8].

Each street light operates in an environment in which the background light can be bright or dark. With respect to the environmental background light, the application scenario has some variants: (i) night (background light is equal to 0.0); (ii) late afternoon (background light is equal to 0.5); and (iii) morning (background light is equal to 1.0). Each street light contains a lighting sensor, but the local brightness also interferes with the sensor measurement.

3.2.1 Problem. During a first simulation, while the background light was always bright, the collection of street lights found a solution that provided a performance, say X+1, during the morning. After the environment changed to the night, the lights' solution was adjusted to deal with this change. However, this new generic solution presents a performance, say X, during the morning, and the street light is unable to return to its previous configuration, as the street light does not maintain different versions of its configuration. This configuration history could enable the street light to maximize its performance during different parts of the day.

In this scenario, is it better to create a street light with a general analysis architecture that can deal with all variants of the background lighting; or a street light that is able to switch its analysis architecture to specialized solutions that were selected for each of the background lighting variants?

3.2.2 Context-aware-based decision. We simulated four scenarios, varying only the background light configuration: (i) always bright; (ii) always dark; (iii) always late afternoon; and (iv) one that dynamically changes the background light at runtime (i.e. sometimes the background light is bright and at other times such as at night the background light is dark).

As proposed in [8], we provided each street light with a neural network to make decisions and we used a genetic algorithm to train the neural network. However, different from the presentation in [8], where one unique neural model dealt with all background light variants, we deployed the simulated street lights with more than one model, which varied according to the background light context. Further, we compared the set of street lights that can use more than one model at runtime against the set of street lights that uses only a single generic model. To generate these models, we trained different neural networks in different contexts, as depicted in Figure 8.

First, we trained a neural network, which consists of five hidden units and uses a binary activation function, putting the set of street lights to interact with the first scenario (always bright). Then, we trained a sigmoid neural network three times (using the scenarios (ii), (iii) and (iv) separately), generating three trained models with particular weight values. In total, we saved four models, such as that one that was generated while the set of street lights interacted only with the first scenario.

3.2.3 Results. As we described, we generated four models by training the neural networks in scenarios with different background lighting contexts: three specific models (one for each background lighting configuration) and one general model (that was trained

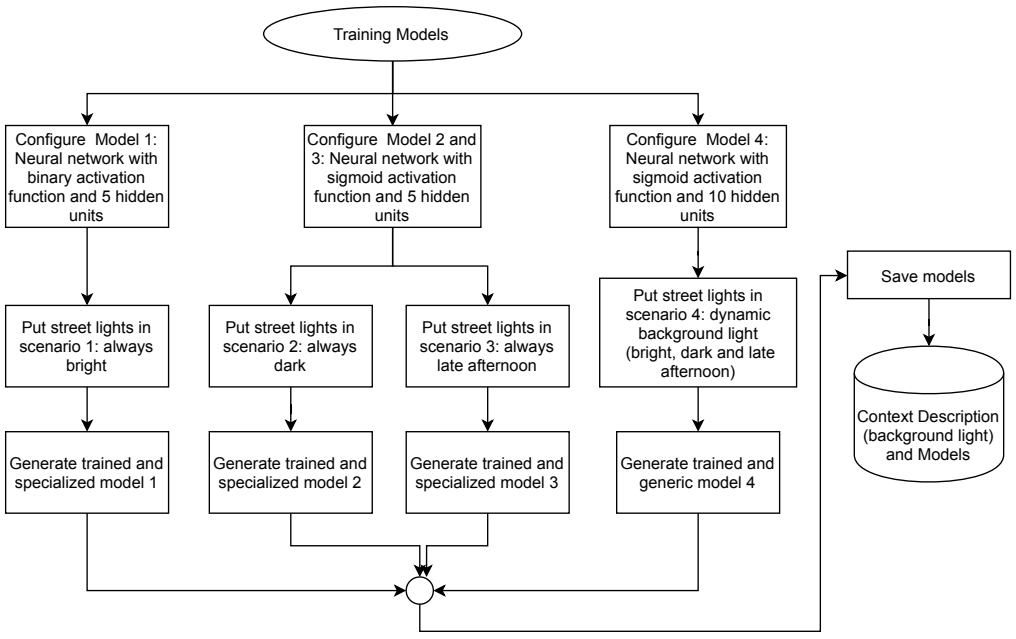

Figure 8: Training step: varying the background light context in a street light scenario to generate generic and specialized ML models.

while the set of street lights interacted with dynamic background lighting). Table 1 shows the performance results that were achieved by the set of street lights while using these models in different contexts. For more details concerning the performance calculations, see [8] .

Table 1: Street lights' performance results obtained while executing different models with different contexts.

ML-based Model		# Interactions	Background Lighting		
			0.0	0.5	1.0
Specific Models	0.0	20	44.85%	x	x
	0.5	20	x	65.49%	x
	1.0	20	x	x	70%
Generalist Model		20	45.42%	62.74%	70%

According to Table 1, the models achieved different results for each part of the day. For instance, the generic model provided the best results for the night and morning. However, the specific model outperformed the generic model during the late afternoon. As the general model is dealing with a more complex scenario, we increased the number of interactions from 20 to 100 to verify if its results could be improved, but they could not.

Accordingly, if we train a unique model to deal with all background lighting variants, as presented in [8], the average performance of the set of street lights in this application scenario is ≈ 59.39%. If the set of street lights is able to select the most appropriate model for each one of the background light configurations, the average performance is ≈ 60.31%. Probably, this difference would be increased if we considered more context parameters.

Therefore, our results show that the best solution for this application scenario is to create a system that is able to use the generic model at night and in the morning, and to switch to the specific model in the late afternoon, as depicted in Figure 9.

However, the difference between the approaches' average performance is less than 1%. Thus, the software engineer must evaluate which approach fits the application requirements better. For example, if the application is critical and performance is the most important requirement, our proposed approach should be considered. If the storage capacity of the application is limited or the system cannot connect to ML web services, finding a general good model may be the most appropriate approach.

4 RELATED WORK

Hernandez et al. [11] discuss the importance of taking the context into account when training a machine learning model and the need of approaches to anticipate the analysis of context changes. Accordingly, they [11] promote the use of incremental learning through reframing, an approach that consists of creating an initial model in a specific context, and then adjusting this model to new contexts. The authors state that their approach can be generalized to many problems and areas in machine learning.

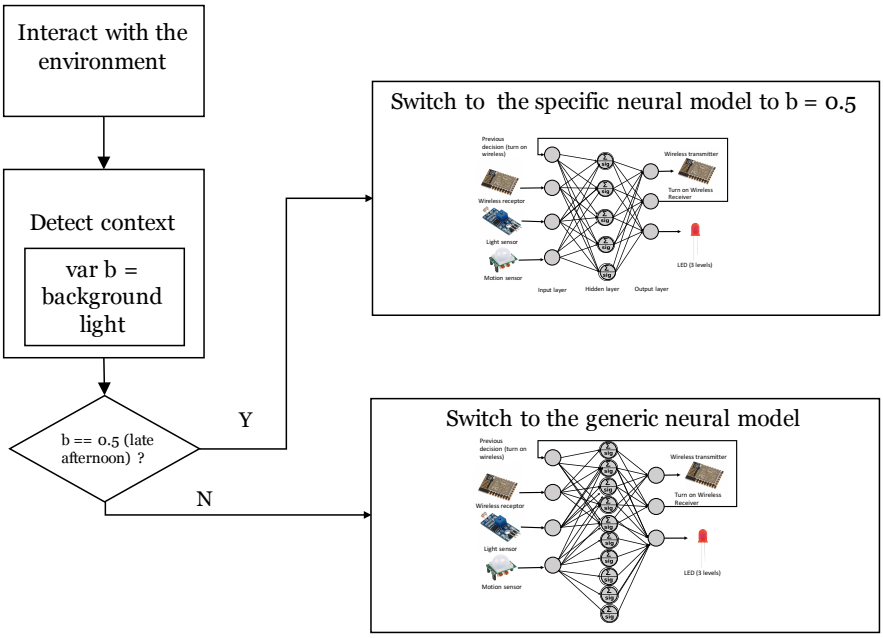

Figure 9: Deployment step: selecting the trained model to use according to the context.

However, Carvalho and Nolfi [4] applied two types of learning approaches to solve a decision task and advocate a different approach. They report a comparison between training a machine learning model at once accessing different environment contexts and training it in a specialized environment before adjusting it to deal with other environmental conditions. Their results show that the general approach outperforms the incremental training approach, as being proposed by Hernández-Orallo et al. [11]. In contrast with Hernandez et al.'s proposed method [11], in our approach, the system is not adjusted to a new context, but it can be totally reconfigured to provide specialized and better solutions for specific context changes.

Mezouar et al. [13] compare local and global (i.e., trained on the whole dataset) effort-aware defect prediction models using 15 projects. They observed that, depending on how the local model is built (i.e. how the subset of the original training data is selected), the local model usually does not perform as well as the global model for the task of predicting defects. Our results show that separating the training data based on the contextual information is promising. In two different experiments, our context-aware approach alleviates bias to the global model. However, we need to perform more experiments, in order to investigate the use of this approach in more application domains, the use of other algorithms, and whether the analysis of context changes can be anticipated.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we described an approach that takes context into account to train and deploy machine learning-based models. To evaluate this context-aware approach, we reproduced two experiments using our proposed architecture: (i) one that describes an air quality prediction task, and (ii) one that describes an autonomous street lights control task. In the first application, we used the month information to characterize the context. In the second application, we characterized the context based on the background lighting information. We selected two completely distinct experiments in order to show that our approach can be generalized to different problems and areas in machine learning. For each experiment, we compared the use of a versatile model that was trained once considering all contexts against a system that is composed of a versatile model and a set of specialized models that were trained for each particular operating context.

Our results showed that our context-aware approach alleviates bias to an approach that uses a unique general and versatile model. However, in both experiments, the error difference between these approaches was considered to be low. Therefore, considering that training a general and versatile model is more cost-effective than training several specialized models for different operating contexts, the software engineer must evaluate which approach fits the application requirements better. Depending on the criticality and the resources (i.e. connectivity, memory) of the application, our proposed context-aware approach should be considered.

Our next step is to select more state-of-the-art experiments and verify if our proposed approach can improve their results. In particular, future work should explore the use of our context-aware approach to alleviate social and political bias. We note that assumptions about the neutrality and objectivity of data may encode serious social and political bias into the results [3, 6]. As an example, it has been shown in [3] that face recognition works better form men with light skin, and error rates for women with dark skin range from 20% to 34%. This result indicates that in some cases, when context (e.g., skin color) is not taken into consideration explicitly in the analysis, significant bias can be introduced.

In addition, we also want to consider more than one attribute to characterize contexts in an application. For example, instead of only using the month information to classify a context in the first experiment (to predict benzene concentration), we could have used the current wind speed, humidity information, and the population of the area including how many factories/cars were nearby. Probably, if we had created more specialized subsets of data, we could have achieved higher levels of improvement.

Finally, we believe that our context-aware approach can further assist a software engineer in achieving a broader understanding of learning strategies and their effect under different contexts.

ACKNOWLEDGMENTS

This work has been supported by CAPES scholarship/Program 194/Process: 88881.134630/2016-01 and the Laboratory of Software Engineering (LES) at PUC-Rio. It has been developed in cooperation with the University of Waterloo, Canada. Our thanks to CNPq, CAPES, FAPERJ and PUC-Rio for their support through scholarships and fellowships.

REFERENCES

[1] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. 1999. Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing*. Springer, 304–307.

[2] Léon Bottou and Vladimir Vapnik. 1992. Local learning algorithms. *Neural computation* 4, 6 (1992), 888–900.

[3] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*. 77–91.

[4] Jonata Tyska Carvalho and Stefano Nolfi. 2017. Exploiting environmental differentiation to promote evolvability in artificial evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 81–82.

[5] David Chappell. 2010. Introducing the Windows Azure platform. *David Chappell & Associates White Paper* (2010).

[6] R Courtland. 2018. Bias detectives: the researchers striving to make algorithms fair. *Nature* 558, 7710 (2018), 357.

[7] S De Vito, E Massera, M Piga, L Martinotto, and G Di Francia. 2008. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B Chemical* 129, 2 (2008), 750–757.

[8] Nathalia Moraes do Nascimento and Carlos José Pereira de Lucena. 2017. Engineering cooperative smart things based on embodied cognition. In *Adaptive Hardware and Systems (AHS), 2017 NASA/ESA Conference on*. IEEE, 109–116.

[9] Adil Farooq. 2016. Modeling and Design of an Energy Harnessing Electric Powered RC Aircraft. *International Journal of Technology and Research* 4, 3 (2016).

[10] Katarina Grolinger, Miriam AM Capretz, and Luke Seewald. 2016. Energy consumption prediction with big data: Balancing prediction accuracy and computational resources. In *Big Data (BigData Congress), 2016 IEEE International Congress on*. IEEE, 157–164.

[11] José Hernández-Orallo, Adolfo Martínez-Usó, Ricardo BC Prudêncio, Meelis Kull, Peter Flach, Chowdhury Farhan Ahmed, and Nicolas Lachiche. 2016. Reframing in context: A systematic approach for model reuse in machine learning. *AI Communications* 29, 5 (2016), 551–566.

[12] Alexandra LâĂŹheureux, Katarina Grolinger, Hany F Elyamany, and Miriam AM Capretz. 2017. Machine learning with big data: Challenges and approaches. *IEEE Access* 5, 5 (2017), 777–797.

[13] Mariam El Mezouar, Feng Zhang, and Ying Zou. 2016. Local versus global models for effort-aware defect prediction. In *Proceedings of the 26th Annual International Conference on Computer Science and Software Engineering*. IBM Corp., 178–187.

[14] Stefano Nolfi and Domenico Parisi. 1996. Learning to adapt to changing environments in evolving neural networks. *Adaptive behavior* 5, 1 (1996), 75–98.

[15] Omer Berat Sezer, Erdogan Dogdu, and Ahmet Murat Ozbayoglu. 2018. Context-aware computing, learning, and big data in Internet of Things: a survey. *IEEE Internet of Things Journal* 5, 1 (2018), 1–27.

[16] Peter Stone. 2000. *Layered learning in multiagent systems: A winning approach to robotic soccer*. MIT Press.

[17] Shimon Whiteson, Nate Kohl, Risto Miikkulainen, and Peter Stone. 2005. Evolving soccer keepaway players through task decomposition. *Machine Learning* 59, 1-2 (2005), 5–30.