Bachelor and Master Theses

Specialization: All Bachelor and Master Specializations

Remarks:

1. All theses must be written in English.
2. Usage of Latex (Beamer) is mandatory.
3. If conclusive results are obtained:
    a. they will be sent for publication at students' symposia, workshops, conferences
    b. teams of students will be encouraged to participate in innovation programs
4. **To work with me:**
    a. you must show and prove outstanding academic record (current GPA greater than 8.50)
    b. you must show disponibility in meeting regularly (weekly or bi-weekly) and tackling research problems
    c. be open to check the business opportunity of the problem you are working on, so you should show interest in entrepreneurship and innovation
5. I also supervise projects proposed by students, but they should be related to my interests and to the topics proposed:
    a. Formal Methods, in particular Static Software Verification;
    b. Automated Theorem Proving, in particular First-Order Theorem Proving;
    c. Software Engineering
    d. Symbolic Computation, in particular Polynomial Algebra;
    e. Distributed Computing, in particular Cloud and Big Data Computing.

Remark! Based on previous experience, many students did not show up, without announcing or having childish excuses, at the meetings in which we discuss the progress of their thesis. To avoid any misunderstandings, I state already that at 3 such happenings you will have to find another supervisor.

Nr	Topic	Observations
1.	Symmetry Breaking for the Cloud Resource Allocation Problem (1-2 theses)	Suppose you want to buy, at the lowest cost, virtual machines (VM) with certain CPU, memory, storage, from cloud providers which are geographically distributed. This is an NP-hard problem which can be formalized as a constraint satisfaction problem and solved using exact algorithms. The problem exhibits symmetries which makes the search for solution to consider already visited solutions, as well as parts of the search tree which are symmetric to already visited parts. The aim of this project is to implement symmetry breaking methods from the paper [1] in the MANeUveR framework (https://merascu.github.io/links/MANeUveR.html) in order to make the problem above amenable to be solved in practice. **Difficulty**: medium/high **Requirements**: *Programming*: Python; *Mathematics*: Computational logic
2.	Graph Neural Networks for combinatorial optimization problems (1-2 theses)	The project aims to apply Graph Neural Networks for optimization problems coming from cloud resource provisioning (see topic 1 above). *Libraries used*: DGL, PyTorch Geometric. **Difficulty**: high **Requirements**: *Programming*: Python; *Mathematics*: Computational logic; graph theory, machine learning, operational research (optimization).
3.	Training Binarized Neural Networks for Traffic Sign Classification (3 theses)	Traffic signs support road safety and managing the flow of traffic, hence are an integral part of any vision system for autonomous driving. While the use of deep learning is well-known in traffic signs classification due to the high accuracy results obtained using convolutional neural networks (CNNs) (state of the art is 99.46%), little is known about binarized neural networks (BNNs). Compared to CNNs, BNNs reduce the model size and simplify convolution

		operations and have shown promising results in computationally limited and energy-constrained devices which appear in the context of autonomous driving. This work aims to extend the paper [2] which presents a bottom-up approach for architecting BNNs by studying characteristics of the constituent layers. These constituent layers (binarized convolutional layers, max pooling, batch normalization, fully connected layers) are studied in various combinations and with different values of kernel size, number of filters and of neurons by using the German Traffic Sign Recognition Benchmark (GTSRB) for training. As a result, we propose BNNs architectures which achieve an accuracy of more than 90% for GTSRB (the maximum is 96.45%) and an average greater than 80% (the maximum is 88.99%) considering also the Belgian and Chinese datasets for testing. The number of parameters of these architectures varies from 100k to less than 2M. The accompanying material of this paper is publicly available at https://github.com/apostovan21/BinarizedNeuralNetwork. The (tentative) *tasks* of this project are: 1. exploring the loss and accuracy: what architecture layers and hyperparameters influence the spikes in the output graph? (thesis 1) 2. investigation of BNNs accuracy when there is a large number of non-linear activation functions (e.g. ReLU) but only fully connected layers (thesis 2) 3. deployment of the machine learning models on real devices (thesis 3). **Difficulty**: high **Requirements**: *Programming*: Python; *Mathematics*: Computational Logic, linear algebra and statistics
4.	Verification of Binarized Neural Networks (1-2 theses)	The trained classifiers from point (3) exhibit issues when it comes to robustness to adversarial attacks. This is an issue since these classifiers are used in autonomous cars which are safety-critical systems. To overcome this issue, application of formal verification can be used. More precisely, given a deep neural network (DNN) and a specification, is there a proof that the DNN satisfies the specification for all inputs? Not surprisingly, the main challenge of applying formal methods to the verification of DNNs is scalability. This is because verification is a non-trivial problem: DNNs are large (high number of neurons and layers) and involve activation functions which are non-linear and non-convex. These make the problem NP-complete. This thesis should check which tools from https://github.com/ChristopherBrix/vnncomp2022_results can be used to check the robustness of the classifiers proposed in [2]. **Difficulty**: medium **Requirements**: *Programming*: Python, C/C++; *Mathematics*: Computational Logic
5.	Designing a half-semester course on using Formal Methods for Software Engineering	This thesis is a support for a half-semester course and lab for teaching formal methods in Software Engineering for computer science students (Bachelor level) that Mădălina will be introducing in her classes. Among the topics discussed should be Hoare logic and weakest precondition, verification of Java Programs (tentative topics). The thesis should contain relevant examples and case studies both at theoretical and practical level (usage of tools). During the development, we will look into other curricula, presented here: https://fme-teaching.github.io. **Difficulty**: medium **Requirements**: *Mathematics*: Computational Logic; *Programming*: Java

References

[1] M. Erașcu, F. Micota, and D. Zaharie, 'Scalable optimal deployment in the cloud of component-based applications using optimization modulo theory, mathematical programming and symmetry breaking', *J. Log. Algebr. Methods Program.*, vol. 121, 2021.

[2] A. Postovan and M. Erascu, 'Architecturing Binarized Neural Networks for Traffic Sign Recognition', 2023.