# Sinopsis

Această lucrare propune un sistem de procesare paralelă în medii hibride CPU-GPU a grafurilor computaționale aciclice, având ca obiectiv principal acela al alocării și execuției eficiente în medii si infrastructuri computationale mobile ce poseda resurse limitate de calcul paralel. Sistemul experimental rezultat din proiect si utilizat intr-o aplicatie comerciala utilizeaza motorul si biblioteca de calcul tensorial TensorFlow - framework-ul pe care Google l-a dezvoltat si lansat Open Source in 2016 – acesta reprezentand mediul de rulare în producție a tuturor serviciilor inteligente realizate de această companie in ultimii ani. Ca aplicație reală, lansata in productie in mediu comercial in 2018, ce utilizeaza principiile execuției optimizate a grafurilor computaționale, vom prezenta si analiza un sistem de analiza, inferenta si predictie a propensitatii de cumparare de produse a clientilor unui mare lant de farmacii. Mentionam ca sistem rezultat in urma dezvoltarii experimentale se bazeaza pe o serie de componente inteligente ce utilizeaza grafuri computationale aciclice ce descriu modele de analiza predictiva proprii domeniului Invatarii Automate. Totodata trebuie menționat faptul că, deși TensorFlow reprezintă cel mai folosit framework de calcul tensorial în domeniul Deep Learning la acest moment atat in mediul academic cat si in mediul comercial, acesta nu este optimizat pentru alocarea eficientă a grafurilor pe sisteme cu resurse computationale limitate si cu atat mai mult pentru procesarea paralelă a mai multor grafuri de tensori în astfel de medii. Astfel, in cadrul proiectului nostru de cercetare-dezvoltare, prin sistemul propus si prezentat in aceasta lucrare rezolvăm într-un mod eficient aceste probleme in contextul dat.

# Abstract

This thesis presents the research and experimental development that targeted the objective of optimally allocating and executing multiple Directed Acyclic Graphs in a parallel fashion on heterogeneous CPU-GPU computing infrastructures that poses limited resource. The proposed experimental system is using an Open Source low-level tensor computation engine, namely TensorFlow [1], a state-of-the-art framework that is able to represents, evaluate and optimize computations based on data-flow graphs. This low-level tensor computation engine has been employed by all Google AI-based production-grade services since 2016. Our proposed system is part of a large commercial product developed by a team of multiple junior and senior data scientists that has the purpose of applying Machine Learning and Deep Learning in particular in order to solve real-life problems in the area of Business Predictive Analytics. The actual real-life problems solved by the commercial product are related to the inference and prediction of customer product preference propensity based on various approaches, tasks that have to be done with mobile computing devices such as notebooks/laptops. Even though TensorFlow has become widely used for Deep Learning tensor-based computation, it is not optimized for highly efficient allocation of tensor graphs on hybrid limited resource environments or for executing multiple tensor graphs in a parallel fashion in such systems. Therefore, in this thesis we will argue that our research and resulted experimental system solves efficiently these proposed real-life issues.

# Introduction

## Context

The whole research started from an actual concrete production-grade commercial product involving the execution of computational graphs for business predictive analytics tasks such as inferring or predicting pharma retail business customer buying preferences propensities.

More precisely, during the implementation of the case study we discovered several shortcomings of the current state-of-the-art tensor processing frameworks, particularly when working with TensorFlow, probably the most advanced tensorial computation and optimization framework currently on the market. Those shortcoming, as it will be further detailed in following chapters, derive from the need of using limited resources for tasks that traditionally and usually require large numeric computational resources both in terms of parallel computing power and memory allocation.

## Problem

As mentioned in the previous paragraphs of our thesis, the main issue we tackled is related to the efficient allocation and execution of tensor computational graphs in limited resources environments and particularly in environments where GPU resources are available although with severe limitations in terms of numerical parallel computing cores and VRAM. TensorFlow is currently the most used and the most advanced tensor computation framework in production-grade systems - as a side note, the most important horizontal industry where TensorFlow is employed is Artificial Intelligence and Deep Learning in particular – however the mentioned tensor computation engine has a few limitations. TensorFlow is not optimised for highly efficient allocation of tensor graphs in hybrid environments where scarcity of memory and computation cores might be an issue – such as mobile computing devices that posses dedicated GPU capabilities. Further, as previously mentioned, in our research and development experiments and in our commercial product development process, we discovered the clear need of executing multiple computational tensor graphs in a parallel fashion with partial (or full) data sharing between the various graph evaluation and optimization processes. We also have to mention that, based on our best knowledge and on our researched further presented in this thesis, there are no available tensor graph computation engines on the market that solve the proposed problem.

## Objectives

The main objective of the project research and development was to create a state-of-the-art approach for computational graph resource evaluation and determination of optimal strategy for execution of the mentioned tensor graphs on limited resources GPU infrastructure. This objective has been based on the actual real-life objective of running on-site experiments without having access to large computational infrastructures due to data confidentiality measures. We can further define our objective with the following three sub-objectives as follows:

1. on-the-fly evaluation of computational graphs required resources;
2. allocation and parallel execution with shared or non-shared data;
3. application of these strategies to a real-life production grade problem.

To further detail the challenge and our approach, a basic use case will be described in detail depicting the real-life problem that has been addressed both directly and indirectly by our work. As previously mentioned, in our scenario the proposed system should be able to execute multiple tensor based graph computations in parallel on a limited resource system such as a mobile computing device. As a result, the system used in this scenario is a usual portable professional computer that offers a basic amount of CPU computation power and an entry level GPU computation device with limited capability of performing parallel numerical computations.

## Solution

In order to fully understand the proposed solution we have to first analyze the scenario and the resulting issues, needs and fine-grained objectives. Finally we will summarize our main solutions approaches based on proposed objectives.

In our scenario we use the system to execute on premises, with no cloud computing support nor local high-performance resources, multiple experiments in the area of predictive analytics for the industrial vertical of pharma retail. In our predictive analytics tasks, we employ deep learning techniques based on tensor-graph computation. The deep learning models are composed of Directed Acyclic Graphs (DAGs) combining multiple architectures such as Fully-Connected Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks and skip connections for optimized numerical gradient flowing. The research and experimentation details of the deep learning models, including the commercial business related key performance indicators, are not entirely in the scope of this thesis and are presented in a separate paper called „*Deep recommender engine based on efficient product embeddings neural pipeline”* - currently in draft stage available on arXiv [2]. Regardless the architectural details of the proposed predictive analytics experiment, the actual challenges and solutions proposed in this thesis can be analysed as a totally separated work entirely related to parallel computation optimization in GPU-CPU environments.

Without going into further details of the deep learning DAGs, we have to analyse their actual running environment – namely the Tensorflow computation engine. Nevertheless, for a quick reference of the proposed directed acyclical graphs architectures, we mention that we use similar techniques and architecture with that of with word2vec [3], doc2vec [4], prod2vec [5], meta-prod2vec [6], GloVe [7] and also sequence-based inference and prediction approaches. Within our scenario (an actual real life example of inferring customer buy preference propensities), we have to process on-site a massive amount of data (namely transactional information) in order to extract and present a inferential and predictive analytics report in near-real-time and on-the-spot. Our scenario has an input dataset of well over 200 million transactions summing over 1 TB of data. Our computational graphs have to analyse all transactional information and compute in parallel graph-maps that describe the relationship of each entity (in our case, can be product, customer, receipt, product category etc.) with each and every other entity.

Based on our research and experimentation we argue that we have developed a solution that soves the above presented problems within the real-life use-case and the solution has the following two main components:

* *Directed acyclical graphs resource evaluation sub-system*. This proposed sub-system has the purpose of evaluating a given DAG resource requirements before the tensor computation engine initiates a evaluation or optimization session. This approach gives us a solution for having enough in-process knowledge in order to allow or not the tensor computation engine to further allocate GPU resources. This particular sub-system basically analyses „off-line” multiple scenarios of evaluation or optimization for a given DAG and generates all the potential resource requirements for each scenario;
* *DAG allocation and execution system*. Based on the knowledge gained by applying the DAG resource evaluation heuristics the execution system uses multi-threaded flow scheduling in order to asynchronously fit the maximum number of heterogenous jobs on the available resources. The DAG jobs execution system bases its heuristics on multi-source knowledge such as: DAG requirements for a given task, external input data resource requirements, inter-DAG data sharing, current available physical resources and finally the existing running jobs

## Results

To shortly summarize our results, we achieved - based on our research, experimental development and deployment of the commercial products - the following results:

* A reliable approach of obtaining tensor computational graph resource requirements before the actual execution of the graph evaluation or optimization within the tensor computation engine;
* A reliable methodology of pre-execution analysis and also managing multiple tensor graph execution sessions, all wrapped around the TensorFlow framework and thus providing support and potential benefits for other academic and commercial projects that use this state-of-the-art framework;
* Ability to run in the final product multiple predictive analytics experiments in parallel that otherwise would have been ran sequentially on a limited resource mobile computing environment. We have achieved comparable overall experiment pipeline execution time to that of running the experiment pipeline in a multi-GPU or in a computing cluster environment. Finally, this product-level ability of our research and development allows the users or consultants that employ the system to provide on-the-spot and almost real-time analyses.

# Requirements Analysis and Motivation

The main motivation of the project has been based on an actual real-life experiment implementation and its particular issues: how to execute multiple jobs of actual computational graph evaluation and optimization on a limited resources mobile computing device by maximizing the available resource usage and actually "squeezing" as much computation and memory allocation as possible. This motivation has generated the list of project requirements that had to address both the scientific and the commercial aspects of the project.

The main motivation and the actual requirements specifications of the proposed researched innovation is not derived from potential customer needs nor from potential use-case scenarios but rather from actual existing customers that are using our innovation in a production-grade system. Our proposes innovation that is presented within this thesis is part of a large scale predictive analytics system that has multiple separate pipelines ranging from our real-life use case - that of real time preparation of market basket analytics based on tensor graph parallel computation, customer retention predictive analytics models (done by other junior data scientist student), recommendation engines, event prediction. All these modules have been researched and developed together with an additional ETL team of two engineering and the overall supervision of a senior data scientist (PhD candidate). It is mandatory to mention that our real-life use-case is the main source for our proposed research and innovation presented in this thesis. Even more, the presented research resulting innovation is totally mandatory for the actually functioning and commercial application of our production grade use-case.

The simplest approach to explaining the proposed innovation motivation is the presentation of the actual process that generated the need for the innovation. The actual real-life production grade use-case of our systems involves, as previously mentioned, the parallel processing of massive amounts of data using tensor computational graphs. Basically, our real-life goal is to provide in a short amount of time the final user/customer of the system with various analysis of the available commercial transactional information, thus constructing a complete relational graph between any two different entities. In order to have even more in-depth information, we can imagine a pharmaceutical retail company that needs a product similarity analysis, product complementarity matching, products composition in encapsulated products, decomposition of compel products, affection-based micro or meta clustering of all its products - all of these features based entirely on customer transactional behavior without any other features. For this hypothesis we have researched and developed our own Deep Learning models (as presented in our research paper [2]) that use NLP-related method in order to generate, with two different approaches, six different products „meta-maps”. As stated, detailed information about our research in the area of Deep Learning for Predictive Business Analytics is beyond the scope of this thesis and can be further analyzed in the mentioned paper, however we will briefly explain the real-life end-customer problem that resulted in the need for our proposed innovation in a step-by-step approach:

1. The end-user of the commercial system is presented with the feature of our system: that of constructing advanced analytics for the so called classic „Market Basket Problem”
2. In order to produce a proof-of-concept in real-time and provide the customer with the confidence that our product delivers promised feature we make the analysis on-the-spot without any cloud-backed resources, without any process of ETL and without any need for customer data export to our internal infrastructure.
3. For the previous mentioned simple proof-of-concept we use fully confidentially transactional data that contains only TIMESTAMP, TRANSACTION\_UNIQUE\_ID and PRODUCT\_ID
4. By employing our own models based on tensor computational graph models we have to create several latent vector multi-dimensional spaces where the products are defined with 128 numbers based on 32-bit floats. The generated latent multi-dimensional spaces can then be used to produce the previously mentioned end-user analytics.
5. In order to run the previously defined experiment we need an environment that will be capable of executing multiple graph computation with multiple data-sources in parallel in order to minimize the experiment running time. Due to the nature of the required tensor computations, basically large matrix multiplications and additions, it is obviously required to employ a simple setup of massive parallel computing that could be used based on mobile computing infrastructure. Thus, we use a basic setup of minimum 386 numerical computing cores provided by a GPU device with a 2 GB RAM. At this point we can define two hypotheses:
   1. We must minimize the execution of our tensor computation by using the array of numerical cores
   2. We must minimize the total execution time of our whole experiment by “squeezing” all the tensor computational graphs and required input data within the available infrastructure

Finally, at this point in our research and experimentation, we arrived at the conclusion that a special GPU parallel computation allocation engine is required. This proposed engine in our envisioned approach had to augment the existing capabilities of the state-of-the-art tensor computation frameworks (TensorFlow in our particular case) with the proposed memory and computation “squeeze” features. This basic requirement resulted in the following main analysis objectives that provided the fundament of our proposed innovation:

1. The requirement to analyze existing state-of-the-art with emphasis on our particular real-life problem with regard to memory allocation strategy for tensor computational graphs;
2. Analysis of a potential engine architecture capable of finding a good strategy for parallel execution of tensor computational graphs based on optimal allocation of GPU resources.