UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS, COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

Artificial scene inference based on efficient parallel execution of directed acyclic tensor graphs

Andrei Ionut DAMIAN (PhD Candidate)

Nicolae Tapus (PhD Supervisor)

2021

Contents

1 Thesis summarization and objectives

1.1 Introduction – artificial scene inference

1.1.1 Real life historical and motivational context

Constructing user interaction interfaces or *graphical user interfaces (GUIs)* is an important aspect of software development no matter the horizontal or vertical target domain nor the deployment environment. Various techniques have been developed since the early days of *GUIs* for the actual designing and implementation of user interfaces for all the platforms balancing both speed of implementation (through the *rapid application development* or *RAD* techniques) as well as quality of graphics and usability. As the systems evolved over the years with generation after generation a new range of needs related to the GUIs appeared: the need to re-think, re-design and re-construct the graphical user interfaces as well as the need to automate behavior - by "automated behavior" we refer the task of sequencing and automating a series of commands that are usual given by a human operator to an application user interface.

The aforementioned needs have been addressed for quite some time with classic techniques such as: re-using the code (if available) and re-designing the graphical user interfaces (*GUIs),* or for the second case of behavior automation applying rule-based actions and events to the 2D coordinate *canvas* of the target user interface – that is "scripting" actions such as "click on coordinates X,Y on the display where there is supposed to be a button". Thus, the rapid application development software tools evolved in order to provide faster and more productive GUI development cycles and, related to automation subject, a new breed of software systems emerged: *robotic process automation* (or RPA) software systems that allow the users to construct automation scenarios where series of UI commands and operations are scripted and executed based on heuristic triggers.

Nonetheless, in all these cases several inherent issues have not been solved such as: (i) re-design of software applications where the source code is no longer available for various reasons; (ii) quickly iteration from graphical designer produced mock-ups to functional interfaces – i.e. from board-designed to actual GUI; (iii) semantic understanding of graphical user interface components (i.e. understanding actual functionality of visual elements) rather then "*blind*" 2D coordinates event generation; (iv) rapid semantic understanding of printed or graphical (on-screen) data forms and information conversion without the limitation of straight

object character recognition. As described in the following sections of the thesis these issues received active attention from both academia and industry and are still important research topics.

1.1.2 The main intuition of the real-life application

The main motivation behind our work has been rooted in the real-life **need to convert legacy applications from old execution environments** – such as legacy desktop database systems or client-server frameworks – **to Cloud-based infrastructures**. This particular objective has multiple roots both in terms of resource management and also from technical perspective. From the resource management motivation perspective, we can enumerate both the costs of contracting and managing the human-based team required for migration software development as well as the opportunity costs related to the time involved in this particular process *(Figure 1 depicts the classical approach for migration and associated direct costs)*. The technical aspects are mostly related to the risks associated with potential buggy modules, configuring and deployment issues and so on.

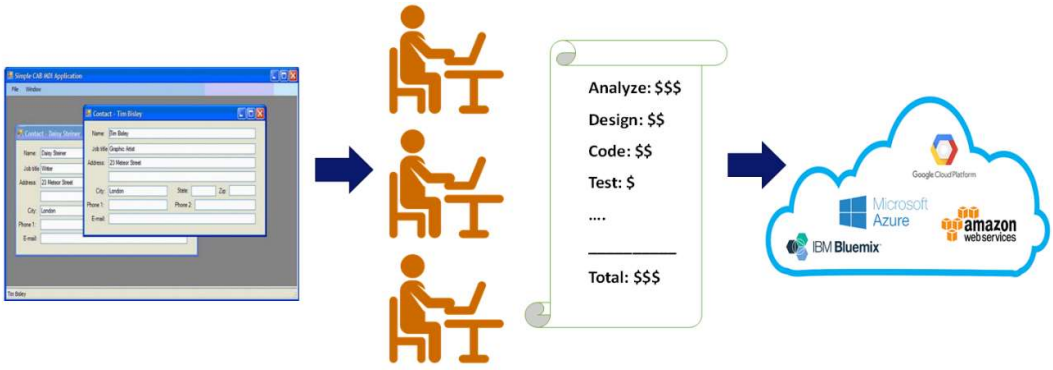

Figure 1 - The classical approach to system migration - the software development team runs a whole development cycle in order to deploy to the new target (maybe) Cloud Computing based environment

Nevertheless, following this initial real-life need identification, several other connected use-cases have been identified in the areas of automation and rapid prototyping: employ

intelligent agents that can automate user-interface human-computer interaction (Robotic Process Automation); transforming a simple (even paper hand-drawn) application user interface mock-up into a functional, designed and scripted user-interface form or screen. To be even more precise, for the user-interface process automation, our target has been that of replacing the need of scripted behavior with intelligent agents. As an example, we can take the case of a scripted agent behavior of clicking a user interface button such as the following one. We can summarize the task in natural language as follows, for simplicity of explanation, although usually it is described in script languages: *"MOVE mouse to this absolute screen rectangle area and perform CLICK operation"* i.e command the mouse to move to a certain location and generate a "click" event on the user interface without human intervention. Now imagine that, while the absolute location remains the same, the button we need to automatically- click moves with its form due to certain circumstances as we will further explain. Starting from this concrete example, our target is that of employing intelligent agents that would recognize in the above example if the ***user-form has been moved*** to another place on the interface screen or that the layout of the input areas has been revamped. Furthermore, we present a list of realistic use-cases based on actual analysis of user needs related to various scenarios:

a) Enrichment and enhancement of simple User-Experience tasks automation or even more complex ones currently provided by RPA systems based on heuristics: arguably the first clear target area of application that is currently lacking this kind of technological advancement to the best of our knowledge is that or Robotic Process Automation (RPA). For this particular case we aim at proposing our research and experimental technology as a logical next step in the enhancement of visual interface automation agents that currently work based on visual rules and heuristics. For a clearer understanding *Figure 2* and *Figure 3* explain the real-life discovered pain-point – while in the first figure the RPA agent is able to complete the assigned task in the second image it clearly fails due to a minor change in the visual UI environment. Only recently (2019) companies in this area advertised new approaches [1] based on Computer Vision for the task of User Experience/Interface components understanding. Nevertheless, this area is still in research and experimentation phase at the moment this report is written.

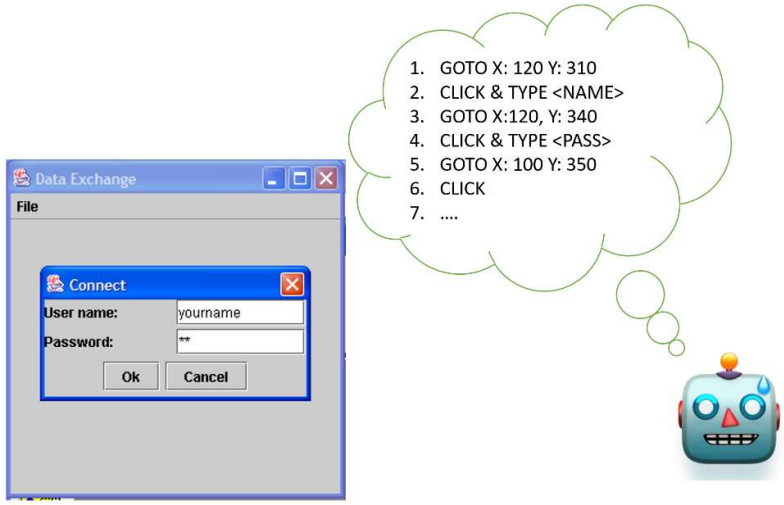

Figure 2 – simple RPA agent pre-programmed based on rules for a specific task goes to certain coordinates and completes the required information. The task is successful

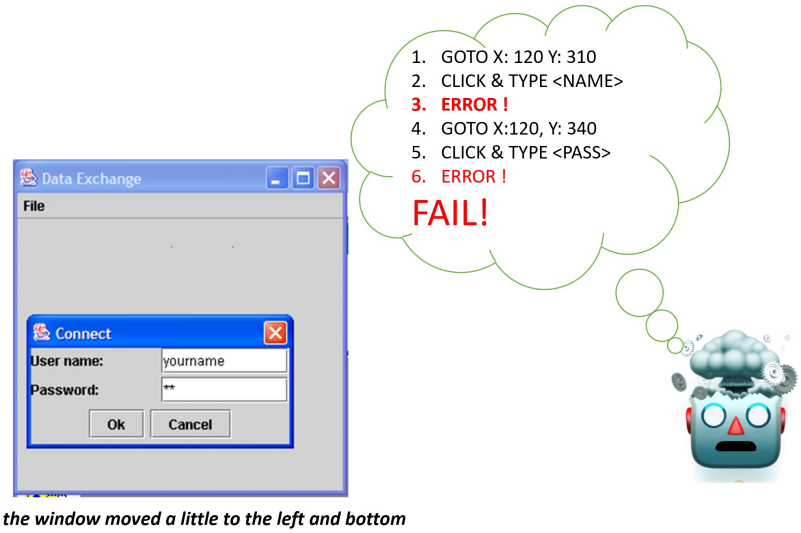

Figure 3 - RPA agent can't complete task as the environment has changed its parameters and the heuristics do not apply anymore (eq: the window has moved from the pre-configured coordinates and the "click" event generates error)

b) Migration, translation, and automated maintenance of legacy software systems such as complex client-server accounting/ERP systems or large-scale banking systems. The potential applications are ranging from the need to translate a legacy

system developed for a particular target operating system desktop graphical environment (a basic example would be develop a new version of the legacy accounting system migrating from a text-based terminal GUI to a modern graphical one) up to the need to quickly and on-the-spot understand user interaction and behavior within complex legacy system.

c) Fast prototyping in unknown or new user experience and user interface programming languages has been constantly an important requirement in front-end development. Although this particular task is closely related to the previous translation and migration one it is nonetheless a different scenario. In this scenario we do not have access to the any source code or already developed graphical user interface. Another potential requirement and constraint of this use case is the requirement for low-to-no-code development – i.e. develop graphical user interface without the prior knowledge of programming languages.

d) Last but not least one of the real-life use cases relates to the advanced understanding and automation of electronic reports, forms and tables. In various both horizontal such as accounting or logistics and vertical domains such as financial sector there is an increasing need for more efficient and streamlined operations through intelligent process automation. Organizations are using either legacy systems or wide-spread systems in order to produce data in various formats and stages of a particular process. Actual example of use cases are companies that generate physical or electronic invoices or transaction reporting documents and then deliver them to their partners, who in turn have the option of manually or automatically extract structured information without the need of an in-place electronic data inter-exchange system. In all these scenarios there is an increasing need for advanced post-processing of the given data without explicit data export and migration – either there is no clear and reliable way of data exporting or the process is to complex for the users that operate the system. For a clearer explanation we will quickly analyze two particular cases:

   i. automated data gathering and processing based on online web-forms. This first case is a straight-forward process of user-interface analysis, inference and translation that, in this scenario, will generate a *script code* output that can range from simple JSON [2] to more complex HTML5 (https://html.spec.whatwg.org/multipage/) and up to PHP

(https://www.php.net/docs.php) or other web-application script languages.

ii. digitization and structured data pre-processing of pre-printed / scanned tables and forms such as monthly financial reports or other similar unstructured data. In this particular scenario the objective is to generate a cross-platform digital representation such as a comma-separated values of a printed document such as a printed spreadsheet that is not available in the digital editable and version-able form.

1.1.3 The overall view

The Artificial Intelligence horizontal achieved great [3] in latest years mainly due to the Deep Learning continuous state-of-the-art improvements and also due to the widely adoption within AI research and development community of GPU-based parallel computing [4] and the proliferation of public dataset libraries [5]. Within the multitude of existing and potential research domains of AI we have to mention the important area of systems development and maintenance automation, still considered by most a "holy-grail" area of research.

Our vision, further presented within this thesis, was to research and develop truly intelligent systems able to analyze user experience video streams from various sources and finally infer real and usable analysis including actual code-level details of those observed interfaces such as the simple example depicted in *Figure 4*. One key element of such systems is that of artificial user-interface scene inference and analysis based on deep learning computer vision systems. During a period of over 2 years we have researched and developed various experiments [6]–[8] that will also be referenced within the thesis with particular emphasis on the research and experiments described in the paper "*Deep Vision Models for Artificial Image Processing*" [8]. Another focus of the past research period has been to analyze and compare our research and experimental work with other similar research and other existing initiatives in this particular field [9].

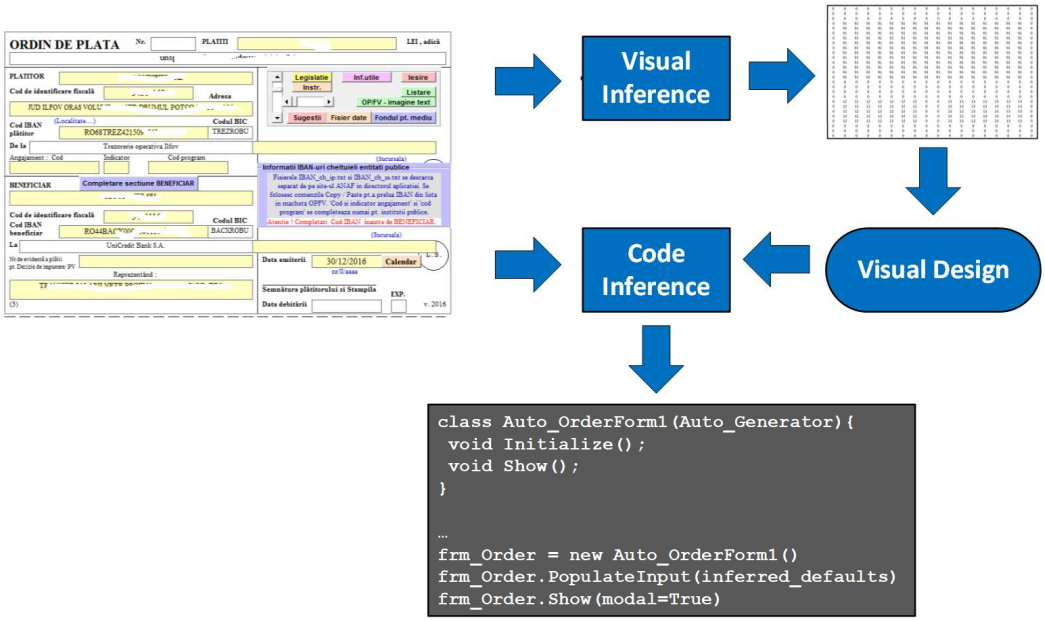

Figure 4 - From a single-form legacy desktop application to a visual design script (such as HTML, JSON,etc) up to the actual source code in a target programming language

As argued in our published work, computer vision models and particularly deep directed acyclic graphs based on convolutional modules are generally constructed and trained based on natural images datasets. Due to this fact, the convolution directed acyclic graph (DAG) models will develop during the training process natural image feature detectors with the exception of the base graph modules that will learn basic primitive visual features. As a result, one of the focus areas of our research and experimentation has been to develop DAG models specialized for synthetic image recognition and train those models on our own experimental datasets. The proposed end-to-end trained models are finally able to infer user-experience functional details of the proposed input synthetic images that can be automatically translated to target operational source-code such as HTML/JavaScript.

In our thesis we will present the current state-of-the-art in two different, yet connected, areas: that of deep learning models for computer vision and the area of GPU-based parallel computation of DAGs for efficient training and production-grade operationalization. Further this we will present the architecture of our whole end-to-end experiment including our early work [6] based on shallow model architectures and the latter Deep Learning based models [8]. A special section will be dedicated to the research and development of our artificial images

dataset that we will publish in Open Source format in order to further benefit the international research community. Following the architecture section, we will continue with the experimentation details and actual details of an online production-grade system.

Finally, we will conclude with a revision of the most important proposed contributions and the actual conclusions of the thesis that include potential multiple real-life applications – from stand-alone implementation of our models up to incorporating the CloudifierNet architectures and pipeline in external RPA (Robotic Process Automation) applications. One important observation is that in our research and experimentation cycles we designed and implemented two different versions of the CloudifierNet namely (v1 referred as *CloudifierNetV1*, v2 referred as *CloudifierNetV2*) and for each version we implemented several sub-version tuning the size of the graph and its computation requirements.

1.1.4 Experimental work vs. real-life application: challenges & expectations

Our final experimental results have been operationalized within a working system prototype that has been deployed live. Currently a team of data scientists and engineers are working on improving the performance of the model pipeline and the fast execution environment.

During our work on this research & development project we have encountered several issues that clearly separate our experimental work results from real-life wide application. Analyzing these individual challenges is important both from the perspective of setting the right expectations of the research ambitions as well as previewing the further work that can be done in order to improve the current results:

a) *The limited domain of application determined by the generation/production of the synthetic training dataset (i.e. the dataset where images/observations are not taken from real-life but rather purely computer generated)*: The research and experimentation task of generating the proposed Artificial Dataset (AD) – namely the images of user interface controls and full user interface screen captures – cannot, by any means, capture all the potential user interfaces variations of any previously or currently available user experience standard or approach. In our thesis, as well as in

the published papers, we emphasize the actual selection of several user interface standard such as legacy Microsoft Windows applications based on MFC, Delphi or other similar development environments. Nevertheless, a universal dataset and thus a potentially universally applicable model pipeline is beyond the scope of our work.

b) *Impractical application of the experimental project results to systems and applications with non-traditional user experience approaches (user interfaces that do not follow classic visual and functional rules):* As a complementary issue to the one previously presented we also have the one regarding the impossibility of our neural model pipeline to "understand" user-interfaces that do not follow common approaches in terms of user-experience flow. One such example is interfaces where say, the "buttons", do not follow any visual pattern the graphical user interface buttons usually have.

c) *Potential need to limit the target development environment to web-based application that do not require complex client-side functionalities*: Finally, following numerous experiments and real use-cases analysis we have concluded that from the multitude of potential target environments (such as mobile-android, mobile-iOS, web, MS-windows, unity, etc.) we will focus on web applications with the classic model-view-controller approach limiting the usage of complex client-side libraries (such as jQuery) to simple and easily deployable features (such as user interaction augmentation features).

1.2 Thesis summarization

In this section we will briefly present the main aspects described within each of the main sections of the thesis with references to the respective individual chapters.

1.2.1 The proposed problem

The motivation behind the thesis and its research and development activities – presented with industry uses cases in ***Chapter 1.1*** - has been that of constructing an efficient and viable approach for the analysis and fine-grained recognition of application user interfaces without any kind of scriptable or programmable heuristics. Detailed elements and background

can be found in the "*Application of artificial scene inference*" chapter. The root of this motivation can be traced to various industry and academic use-cases such as the need for self-adapting robotic process automation or the need to convert legacy applications to new infrastructures such as Cloud Computing without the availability of explicit initial source code. Other real-life industry use-cases worth mentioning are recognition of completed electronic forms or electronic form templates or data extraction or data entry or fast-prototyping of modern user interfaces without the need of software design tools such as Rapid Application Development tools.

In order to have a more concrete view and better understanding we will breakdown a couple of use-cases into problem definition, proposed solution hypothesis and the actual industry application:

i. Automatic computer visual recognition of application user interfaces without access to the pre-defined rules or heuristics
   - **Problem**: Given screen captures of legacy applications user interfaces generate collection of interface artefacts, locations and purposes.
   - **Hypothesis**: Using highly optimized numerical computation tensor graphs deployed on GPU compute we detect and localize each of the user interface objects
   - **Application**: Robotic process automation benefits from this use-case by dropping the need for scripted rules and heuristics thus reducing implementation and configuration time and drastically increasing overall viability and dimension/location/aspect invariability

ii. Quick prototyping of functioning user interfaces based on old visual interfaces or simple mock-ups
   - **Problem:** Given screens of user interface mockups, legacy applications screen-shots or simple human create hand-drawn interfaces (such as the one depicted later in *Figure 25*) generate full user-interface script code
   - **Hypothesis:** Using highly optimized numerical computation tensor graphs deployed on GPU compute analyze input images and directly generate sequences of text-code (script)
   - **Application**: From-drawing-to-code quick-prototyping of user-interfaces directly by non-programmers – thus without programming language prior

knowledge - or visual designers and source code generation for a target Cloud app platform based on the old legacy user interface screens;

1.2.2 Related work chapter summary

The analysis of current state-of-the-art and how it relates to our work is presented in ***Chapter 2*** of the thesis. Multiple areas of hot research topics are strongly connected to our however two different areas stand-out as the main ones: GPU based approaches for numerical computing efficiency optimization and advanced architectures for construction of directed acyclical graphs for deep computer vision and natural language processing.

The mainstream adoption of GPU-based devices as well as the recent fast advancement and the new family additions (such as the more recent TPUs - Tensor Processing Units) has enabled both the academic and the commercial community to process large quantities of data using highly parallelized numerical algorithms. Discrete convolution operations, multi-branch numerical graph computation, *Transformer*-like [10] architectures are just few examples where GPU/TPU based massive parallel numerical computation shine. In our work, a few particular areas and individual components have received special attention with regard to parallel numerical compute optimization: the new proposed approach for multi-gated sub-graphs and the overall artificial scene recognition directed acyclical graph. Although GPU kernel construction is a subject highly abstracted by tensor computation and optimization frameworks such as *Tensorflow* [11] or *PyTorch* [12], we did mention the techniques and sample approaches in our related work.

Deep learning architectures resulted from the latest researched state-of-the-art are the second main area of related work presented in ***Chapter 0*** as well as in ***Chapter 2.2*** . First of all, we inventoried the known classic approaches in Computer Vision focusing on more recent approaches based on deep directed acyclical graphs. Although the main presented deep vision architectures directly relate to various known problems and use-cases, the main focus of our experiments and thus the presented results are focused on graphical user interface scene inference (object localization and detection) as well as pixel level segmentation.

A secondary focus has been directed in the area of generative models for source code sequence generation. This has been done both in conjunction with the research on the subject

of scene segmentation and image captioning approaches as well as analyzing the modern trends in neural natural language processing.

Last but not least, within this area of research one important focus subject has been that of finding optimal and efficient architectures for tensorial directed acyclical graphs. More specifically an important aspect of the work has been that of finding innovative approaches for self-tuning of graph architectures that leverage parallel numerical computing instead of classic approaches such as exhaustive or random search of graph architecture option space (grid-search). In this particular area, papers and past experiments on self-learning gating mechanisms have received special attention in ***Chapter 2.5*** and briefly in ***Chapter 2.2.2***.

To summarize the main areas of related work we have the following:

i. Highly efficient **tensor graph evaluation based on GPU** numerical parallel compute
ii. Deep vision architectures for **artefact localization and segmentation**
iii. Generative methods for **image-to-code-sequence** translation
iv. **Graph hyperparameter self-learning** based on actual graph parameter tuning using optimization objectives

1.2.3 Architecture chapter summary

The whole research and experimentation processes has been based on a cyclic approach and step-by-step advancement starting from simple approaches into deeper and more complex solutions for the proposed objectives. This process led to two types of results as described in ***Chapter 3.1***: the domain specific results and deliverables as well as results that can be applied cross-domain and have already been applied to several real-life industrial use-cases. The detailed design process and the architectural approach with all their details are fully described ***in Chapter 3*** of the thesis. The initial steps of the research and experimentation process have been entirely based on basic "shallow" (i.e. machine learning approaches that do not have hidden graph layers) approaches of using regression models in parallel numeric computation environment in order to gauge the option of having ensembles of simple models. Basically, the initial "baseline" approach, based on simple machine learning models, has been that of employing parallel dot-product computations of input space – the RGB image representation of the user-interface – with weight matrices representing potential user interface primitives, all

this using 2D sliding-window mechanisms. More concretely, we leveraged GPU parallel numerical compute capabilities in order to compute multiple 1-vs-all hypotheses for all potential regions of the target user interface image.

Following this rather naive approach to user-interface screen-shot scene inference, we started developing more complex approaches based on directed acyclical graphs with discrete and separable convolution modules – this architectural approach being fully presented in ***Chapter 3.6***. In this process of refining architectures our objective has been that of constructing a well optimized and balanced class of deep directed acyclical graphs in terms of performance vs costs and energy consumption. Thus, we researched the potential benefit of constructing self-learnable sub-graphs based on learnable gating mechanisms that are explained and formalized in ***Chapter 3.3***. This innovative approach would allow the directed acyclical graph to adapt its own operation and data-flow structure, based on the overall optimization objective. Several important aspects regarding the convolutional module based directed acyclical graph architecture are presented further in ***Chapter 3*** such as the analysis of transfer learning options we explored in ***Chapter 3.4***, comparison between training with the artificial data versus the hand-drawn human generated images in ***Chapter 3.5*** and details of optimization process in ***Chapter 3.7***.

In order to synthetize the main principles used in our experimental architecture design we can summarize the following:

i. A set of **basic goals and principles** has been laid-out from the verry beginning such as using efficient parallel numerical computation mass-market infrastructures based on GPU
ii. Experiments have been **started from low-level, low-complexity** approaches to setup clear baselines
iii. The **complexity has been added gradually increased** and more challenges have been tackled iteratively
iv. Finally, the self-learning graph architecture based on self-gating mechanisms allowed the generation of our DAG family

1.2.4 Notable research results

There are three different areas where our work has produced validated results with the important mention that some of them are currently used in production grade implementations of various industries uses cases. Probably the most important and industry-relevant research result is the proposal for an innovative approach of tuning directed acyclical graphs hyperparameters based on self-learning instead of classical exhaustive or random grid searching in hyperparameter spaces, in essence the **Multi** *self*-**Gating** mechanism. This particular innovation described in detail ***Chapter 3.3***, currently in use in a series of production grade systems in the area of predictive analytics - demand forecasting and event prediction – thus demonstrating cross-industry application. The domain agnosticism of this proposed innovation shows great promise and further work is planned in this area in the next period. Moreover, in this area of research particular focus has been directed towards green computation – i.e. **energy efficiency** – as well as **experiment explain-ability** - one of the hottest topics in today machine learning landscape.

The second general area of research where our results have been validated either through published research and experimentation and/or industrial experimental development is that of artificial image-scene (e.g.: user interface screen-shot) analysis and source code (script) generation. In this particular area we have developed – the third main result - a new dataset that is publicly available for experimentation as well as methodology for potential expansion of this proposed dataset. Beside the dataset our work has been focused on finding efficient directed acyclical graph architectures that would successfully reach both the objectives of segmenting user interface pictures as well as generate basic user interface design source code (or script).

As a summarization of the most important research results, we have the following three main points:

i. Proposal for the new **Multi Gated** subgraph **Unit** that enables self-learning of graph topology structure (reconfiguration of nodes and arcs)
ii. Efficient discrete convolutional module **architecture for artificial scene inference**
iii. Publicly available artificial scene dataset – a ***ImageNet* for the user interfaces**

1.2.5 Experiments

In terms of experimentation, as it will be clearly presented throughout ***Chapter 3*** and more specifically in ***Chapter 4***, we have had two different types of experiments: those that are directly related to the main objective of the thesis as well as experiments, both purely academic or actual industry systems, related to the proposed innovations that are agnostic to our proposed goals. To quickly mention the main industry applications we need to mention that we have both deep vision related use cases in medical domain as well as predictive analytics use-cases.

For the first category of experiments - those related to user interfaces screen inferences – our work, presented in the results related ***Chapter 4.3***, has been focused on correctly assessing the performance of the proposed directed acyclical graph architectures on the collected data as well as iteratively increase the quality and quantity of *CloudifierNet* dataset. Beside measuring performance in user interface artefacts detection and localization a secondary area of experiments targeted the scaling and efficiency of computation. More concretely we experimented with different graph sizes – from a minimal size up to a 2x, 3x expansion of graph nodes and arcs - as well as different compute capabilities such as scaling from a mere 256 numerical core embedded infrastructure to over to 4000-8000 numerical core infrastructures commonly found in modern GPUs.

In this particular class of experiments and particularly related to the proposed *CloudifierNet* dataset we also created a series of experimental applications with various development tools with the main purpose of generating visual user interfaces as well as user interface design source code. More concretely, we designed both *win32* based visual applications as well as portable POSIX compliant visual applications without any complex process or business logic, all with the sole purpose of generating artificial scenes for our dataset – images (screen shots) of potential user interfaces for various cases.

Aside from all previously mentioned artificial scene inference related experiments we pursued a secondary experimentation pipeline for the proposed project-agnostic innovations presented initially in ***Chapter 3.3*** and evaluated in ***Chapter 4.4***. A particular focus in this area has been that of experimenting with various directed acyclical graph architectures with or without the usage of our proposed *Multi Gated Unit*. The main objective of these experiments has been that of testing the performance improvement on simple classification tasks (such as classifying MNIST single channel images of hand written numbers) that a classical directed

acyclical graph receives after being augmented by our *Multi Gated Unit.* The secondary objective of the experiments in this area has been that of self-explain-ability: automatic extracting of gating information and performing – if possible – gate pruning operations thus decreasing the number of matrix multiplications required to process the whole computational graph. This proposed pruning approach would allow obtaining non self-gated subgraphs from modules where the gates are fully closed or opened.

1.2.6 Main personal contributions summarization

Directly related to both the experimentation results and the underlying proposed directed acyclical graph architectures and pipelines, in ***Chapter 5*** we have the following list of main research and innovation contributions as well as practical applications deployed in real-life scenarios:

i. ***Multi Gated Unit***: Proposal of a new innovative approach for finding optimal graph structure (self-learning graph hyperparameters) directly through the minimization of the task's objective function. This particular contribution is domain-agnostic and has been tested both in *deep vision* related tasks as well **as successfully implemented in production-grade predictive analytics systems**. The two main results of this particular research contribution ca be summarized as follows:
    a. Eliminating the need for grid-search approaches that would require running up to millions of experiments on parallel compute infrastructure and thus drastically reducing the **carbon footprint** of the graph architecture search procedure
    b. Self-explain-ability of the model inner structure based on the gate activations within each Multi Gated Unit for the whole graph. Another view of viewing this finding is that of having a method for defining a unique structure (hyper-parameters) for each individual node of a deep directed acyclical graph.

ii. ***CloudifierNet dataset***: Open Source publication of a novel dataset that enables research and experimentation in the area of automatic recognition of user-

interface content. The dataset can be used in a wide range of experiments such as: training agents to recognize user interface elements in order to simulate/generate application messages, generate basic functionality source code for layout design or even construct generative approaches (such as *generative adversarial networks*) that could potentially automatize the design process beside the source code creation.

iii. ***CloudifierNet architectures***: The proposed directed acyclical graphs designs are based on incremental improvements on top of several state-of-the-art architectures while also leveraging the multi-gating modules.

iv. ***Real life applications for medical domain***: Using the proposed CloudifierNet architecture, in the past 2 years, we managed to develop and propose production grade systems in the area of oncology such as:
   a. Full auto-tuned system based on proposed CloudifierNet architecture for dermatologists and inference of severe dermatology lesions
   b. Application able to perform colposcopy (or *cervigram*) analysis for oncological gynecology in order to detect potential cervix lesions and their severity

1.2.7 Further work

In ***Chapter 6*** of the thesis a series of improvements are proposed, further work that is currently in research and experimentation phase for two different areas. One of the proposed areas of further work is specific to the task artificial image scene processing while the second general direction of improvement is related to hyper-parameters self-tuning.

In the area of artificial scene recognition, we have several clear directions of planned further work as well as potential directions of exploration, all described in ***Chapter 6.2***, ***Chapter 0*** and finally ***Chapter 6.4***. The main driver of the planned-ahead work in this general direction is strongly tied both to the real-life use-cases that we target as well as the academically related objective of improving the *CloudifierNet* dataset.

In the area of hyper-parameter self-tuning our goals, described in ***Chapter 6.8***, are divided between further improving the energy efficiency gains as well as advancing the self-explain-ability and self-pruning of the proposed *Multi Gated Unit*. We strongly believe that a more reliable and universally applicable *Multi Gate Unit* will encourage both academic and industry research teams to employ it rather than apply classic hyper-parameter grid-search space exploration. No matter the target domain where the self-tuning of hyperparameters is applied, the carbon print reduction of the optimal graph architecture search will be dramatically reduced when generating a single self-learnable architecture rather than performing searching operations in hyperparameter space. Although a single classical deep graph optimization process might have a much lower energy cost than that of a self-tuned one – due to the multitude of added gating mechanisms computations in the Multi Gated Unit – using the later one involves a ***single*** full graph optimization iteration.

In order to clearly summarize our proposed further work, we can enumerate the following objectives based on the proposed Chapter 6 of the thesis:

i. Advanced user-interface artificial scene recognition including inference from videos and inference of process behavior;
ii. Further improvement iterations of the *CloudifierNet* dataset and the creation of a academia and industrial community around this dataset;
iii. Finally, the improvement of the *Multi Gated Unit* both from the perspective of successful application in various use-cases as well as advancing the self-explain-ability and computation reduction using self-pruning mechanisms.