**Nr.: 139/30.12.2016**

**Raport stiintific**

**de cercetare-dezvoltare in cadrul Cloudifier SRL**

|  |  |
| --- | --- |
| **Nume proiect** | Platforma de migrare automatizată în cloud a aplicațiilor și sistemelor informatice clasice cloudifier.net |
| **Beneficiar** | CLOUDIFIER SRL |
| **Cod MySMIS** | 104349 |
| **Nr. iregistrare** | P\_38\_543 |
| **Director Proiect** | Andrei Ionut DAMIAN |
| **Activitate conform planului de proiect** | 1. Activităţi de cercetare-dezvoltare (cercetare industrială şi/sau dezvoltare experimentală) - 1.1 State-of-the-art |
| **Echipa de cercetare-dezvoltare** | Andrei Ionut DAMIAN  Octavian BULIE |
| **Livrabil** | Raport State-of-the-Art / 74 pag |

Contents

[1 Abstract 3](#_Toc470948471)

[2 Analiza contextului cercetarii 4](#_Toc470948472)

[2.1 Acronime 4](#_Toc470948473)

[2.2 Obiectivele proiectului 5](#_Toc470948474)

[2.3 Contextul proiectului 6](#_Toc470948475)

[2.4 Sumarul activitatii de cercetare si rezultatele 8](#_Toc470948476)

[2.5 Alte remarci 10](#_Toc470948477)

[3 Rezultatele cercetarii 11](#_Toc470948478)

[3.1 Problematica concreta analizata 11](#_Toc470948479)

[3.2 Ecosistemul Cloudifier.NET 15](#_Toc470948480)

[4 Doua directii de abordare avansata a cercetarii 16](#_Toc470948481)

[4.1 Tehnici bazate pe modele avansate Shallow Machine Learning 17](#_Toc470948482)

[4.1.1 Paralelizarea masiva a modelelor superfiale de invatare 17](#_Toc470948483)

[4.1.2 Framework-ul state-of-the-art XGBoost 21](#_Toc470948484)

[4.1.3 Modele proprii testate in Eigen 30](#_Toc470948485)

[4.2 Tehnici bazate pe Retele Neurale Adanci 61](#_Toc470948486)

[4.2.1 Utilizarea CNN in segmentarea semantica 61](#_Toc470948487)

[4.2.2 Utilizarea LSTM pentru sisteme expert de tip Bot 61](#_Toc470948488)

[4.2.3 Framework-ul state-of-the-art TensorFlow 62](#_Toc470948489)

[4.2.4 Framework-ul state-of-the-art Keras 63](#_Toc470948490)

[5 Anexa – Rapoarte lunare 64](#_Toc470948491)

[5.1 Raport stiintific lunar 1 64](#_Toc470948492)

[5.2 Raport stiintific lunar 2 68](#_Toc470948493)

[5.3 Raport stiintific lunar 3 71](#_Toc470948494)

# Abstract

In prezentul document este descris rezultatul etapei de cercetare industriala aferenta Activitatii 1 de cercetare industrială şi/sau dezvoltare experimentală, sub-activitatea “State-of-the-Art”. Cercetarea industriala realizata in aceasta etapa a constat in revizuirea stadiului actual al tehnicii/tehnologiilor atat in domeniul Cloud Computing / virtual desktop dar mai ales in domeniul recunoasterii si segmentarii semantice a imaginilor prin algoritmi avansati de inteligenta artificiala, domeniu de cercetare ce a luat o deosebita amploare in ultimile 12 luni.

Scopul analizei, recunoasterii, segmentarii semantice si a constructiei unei harti a imaginilor este de incorporare a acestor tehnologii avansate de inteligenta artificiala (Machine Learning) in motorul de migrare automatizata a aplicatiilor in mediul cloud. Intuitiv acest mecanism functioneaza similar unui actor/observator uman care analizeaza o imagine dupa care reproduce intr-un alt mediu din memorie elementele principale ale imaginii respective impreuna cu pozitia, forma si functionalitate lor in contextul analizat initial.

# Analiza contextului cercetarii

## Acronime

**Machine Learning** = domeniu de cercetare-dezvoltare din cadrul Inteligentei Artificiale, aflat la fundamentul acesteia, prin care un sistem computerizat este capabil sa invete automatizat din propriile greseli cu sau fara supervizare sau orice forma de factor uman

**Desktop computing** = mod de lucru in care aplicatiile utilizate sunt rulate exclusiv pe o statie de lucru sau laptop fara a exista acces la o baza de date centralizata sau la sisteme online

**Web- based computing** = mod de lucru in care aplicatiile sunt rulate in cadrul unui server de web iar utilizatorii acceseaza functionalitatile respective prin intermediul browser-elor de internet

**Legacy** = Aplicatie sau modul dezvoltat in tehnologii invechite si/sau perimate care ruleaza in mediu de tip desktop computing

**Client-Server** = mod de lucru cu masive de date prin care acestea sunt centralizate pe un server de baze de date relationale

**Segmentare semantica** = analiza unei multimi, imagini, etc si identificarea elementelor componente ale acesteia

**Cloud computing** = tehnologie si mediu de rulare a sistemelor informatice prin care toate datele si functionalitatile aplicatiilor se acceseaza prin intermediului internetului si a aplicatiilor de tip web-based

**Bot** = aplicatie care simuleaza un interlocutor cu care utilizatorul comunica in limbaj natural (roBot) pentru rezolvarea unui set de probleme analizate cu ajutorul algortmilor de tip Machine Learning

**Python** = limbaj de nivel inalt construit in special pentru mediul stiintific ce include capabilitati avansate pentru dezvoltare atat la nivel de experimentare cat si la nivel de productie

## Obiectivele proiectului

In conformitate cu Cererea de finantare aferenta contractului de finantare nr 98/09.09.2016 obiectivul proiectului „PLATFORMA DE MIGRARE AUTOMATIZATA IN CLOUD A APLICATIILOR SI SISTEMELOR INFORMATICE CLASICE- Cloudifier.NET” este cercetarea, dezvoltarea si punerea in functiune in mediul comercial a produsului platforma inovativ Cloudifier.NET, ce se adreseaza domeniului tehnologiilor informatiei si comunicatiilor. In cadrul acestui obiectiv mentionam si intentia de diseminare publica partiala a rezultatelor proiectului sub licenta European Public License.

Obiectivul proiectului “PLATFORMA DE MIGRARE AUTOMATIZATA IN CLOUD A APLICATIILOR SI SISTEMELOR INFORMATICE CLASICE - Cloudifier.NET” raspunde uneia dintre prioritatile stabilite de Comisia Europeană prin Agenda Digitala 2020 si anume stimularea si facilitarea comertului electronic la nivelul tarilor membre. Acest proiect va contribui major la sporirea utilizarii, calitatii si a accesului la tehnologia informatiei si comunicatiilor atat la nivelul local al Romaniei cat si la nivelul comunitatii de viitori potentiali utilizatori din intreaga Uniune Europeana iar implementarea cu succes a platformei va duce la sporirea contributiei sectorului TIC pentru competitivitatea economica a Romaniei.

De asemenea, proiectul are ca obiectiv integrarea federalizarii cu conceptul de virtual desktop online, oferind astfel utilizatorului posibilitatea de a integra toate aplicatiile pe care le utilizeaza in mediul online – aplicatii de tip Cloud– intr-un singur spatiu virtual, in care sa dispuna de servicii de securizare si confidentialitate avansata a informatiilor.

La finalizarea implementarii produsul final va functiona ca o platforma de tip desktop online, bazata pe tehnologii de tip Cloud Computing, cu functionalitati multi-scop, care va permite sustinerea proiectelor inovative din domeniul tehnologiei informatiilor si comunicatiilor si va facilita dezvoltarea sustenabila a acestora.

## Contextul proiectului

Produsul Cloudifier.NET revendica prin unicitatea conceptului sau o serie de functionalitati total inovative, ce nu se regasesc la produsele comerciale deja existente in piata. In procesul de analiza preliminara a stadiului actual al tehnologiei (state-of-the-art), s-au luat in considerare o serie de produse comerciale relativ similare, pentru a se face o paralela intre acestea si Cloudifier.NET si a se determina detaliile avansului tehnologic dincolo de stadiul actual al tehnologiei (advances beyond state-of-the-art).

Implementarea proiectului „Platforma de migrare automatizata in Cloud a aplicatiilor si sistemelor informatice clasice- Cloudifier.NET” va permite:

* Sustinerea proiectelor inovative din domeniul tehnologiei informatiilor si comunicatiilor;
* Va facilita dezvoltarea sustenabila a acestora, prin oferirea accesului la o platforma de tip comunitate, in care sa poata gasi si regasi aplicatii si sisteme la cerere, utilizand astfel exclusiv structura de costuri de tip OPEX, fata de structurile clasice de tip CAPEX;
* Sprijinirea companiilor mari prin servicii “cost-effective” pentru migrarea de la aplicatii legacy- (aplicatii dezvoltate prin metode clasice de programare si implementare de tip desktop sau client-server, ce utilizeaza resurse bazate pe cheltuieli de capital, cum ar fi echipamente de calcul locale, licente locale, s.a.m.d - spre deosebire de modelul de aplicatii bazat pe tehnologia Cloud Computing care utilizeaza resurse la cerere scalabile, elastice si bazate aproape exclusiv pe modelul de cheltuieli operationale OPEX)" - la aplicatii in Cloud;
* Sprijinirea atat a mediului IMM, cat si a utilizatorilor privati, in vederea accesului la un mediu de tip spatiu virtual, personal de lucru online, in continua dezvoltare.

Unificarea avansului tehnologic adus de subplatforma avansata de translatare bazata de Machine Learning a aplicatiilor clasice in medii de tip Cloud cu avansul tehnologic propus de subplatforma de federalizare, brokeraj de date si spatiu virtual de lucru (virtual desktop), face ca platforma sa ofere o arie de inovare deosebit de generoasa, acoperind mai multe nevoi orizontale in domeniul tehnologiilor informatiilor si comunicatiilor, ce au impact asupra unor arii multiple tehnologice si variate industrii.

Cele doua subsisteme principale ale platformei Cloudfier sunt:

1. subsistemul bazat pe Machine Learning de translatare inteligenta automatizata a aplicatiilor clasice desktop in aplicatii online in mediu de tip Cloud computing cu accent pe trecerea de la modele de sisteme informatice bazate pe CAPEX la modele de sisteme informatice bazate pe OPEX;
2. subsistemul de federalizare a aplicatiilor de tip Cloud Computing provenite din surse multiple in vederea realizarii unei platforme de tip spatiu de lucru personal virtual online (online personal virtual desktop).

In concluzie directiile principale de utilitate ale platformei propuse sunt axate pe trei mari categorii distribuite in doua zone de inovatie dupa cum urmeaza:

1. Inovare in domeniu federalizarii platformelor, migrarii datelor si spatiilor personale virtuale:
   1. broker de servicii de Cloud pe care se inregistreaza furnizorii de servicii noi si inovative de Cloud;
   2. agregator si federalizator de servicii de Cloud prestandardizate
2. Inovare in domeniul migrarii aplicatiilor construite pe principiile clasice ale sistemelor informatice catre noile paradigme tehnologice definite de Cloud Computing:
   1. sistem inteligent bazat pe tehnici avansate de Machine Learning destinat traducerii applicatiilor clasice desktop sau aplicatiilor client-server in aplicatii de tip Cloud Computing;
   2. provizionarea automatizata a aplicatiilor translatate in mediul platformei inovative

## Sumarul activitatii de cercetare si rezultatele

Activitatea de cercetare a stadiului actual al tehnologiei s-a axact atat pe o serie de proiecte depuse in cadrul apelurilor de proiecte de cercetare beyond-state-of-the-art din cadrul Horizon 2020 dar mai ales pe lucrari publicate in cele mai renumite jurnale stiintifice cum ar fi:

* Journal of Machine Learning Research, www.jmlr.org (ISSN 1533-7928)
* International Journal of Neural Systems, ISSN:0129-0657
* IEEE Transactions on Pattern Analysis and Machine Intelligence, ISSN:0162-8828
* IEEE Transactions on Neural Networks and Learning Systems, ISSN:2162-237X
* Machine Learning, ISSN:0885-612
* International Journal of Intelligent Systems, ISSN:0884-8173
* Expert Systems with Applications, ISSN:0957-4174

De mentionat este faptul ca au fost analizate in mod particular lucrari ale celor mai cunoscuti cercetatori din domeniul inteligentei artificiale ca:

* Yoshua Bengio: http://www.iro.umontreal.ca/~ben...
* Geoffrey Hinton: http://www.cs.toronto.edu/~hinton/
* Alex Smola: http://alex.smola.org/
* Andrew Ng: http://ai.stanford.edu/~ang/
* Alex Krizhevsky: http://www.cs.toronto.edu/~kriz/
* Ilya Sutskever: http://www.cs.toronto.edu/~ilya/
* Andrej Karpathy: http://cs.stanford.edu/people/karpathy/
* Francois Chollet: https://blog.keras.io/author/francois-chollet.html
* Chris Meek: http://research.microsoft.com/en...
* Hugo Larochelle: http://www.dmi.usherb.ca/~larocheh/index\_en.html
* Ian Goodfellow: https://en.wikipedia.org/wiki/Ian\_Goodfellow
* Aaron Courville: https://aaroncourville.wordpress.com/
* Leo Breiman: http://www.stat.berkeley.edu/~br...
* Andrew McCallum: http://www.cs.umass.edu/~mccallum/
* Chris Meek: <http://research.microsoft.com/en>...
* Trevor Darrell: https://people.eecs.berkeley.edu/~trevor/
* Jonathan Long: http://people.eecs.berkeley.edu/~jonlong/

Din lucrarile analizate au fost extrase si analizate elemente cum ar fi:

1. Modelele de reconstructie arhitecturala si regenerare cod sursa
   1. Modele si metode de analiza a segmentarii ecranelor din cadrul aplicatiilor legacy bazate pe algoritmi clasici interativi si modele de Machine Learning fara retele neurale adanci
   2. Modele si metode de segmentare semantica cu ajutorul retelelor neurale adanci convolutionale a componentelor din cadrul ecranelor aplicatiilor legacy
2. Modelele de migrare aplicatie
   1. Utilizarea retelelor neurale recurente si in particular a LSTM (Long Short Term Memory) pentru realizarea de sisteme expert de tip bot care sa fie capabile de a asista utilizatorul in procesul de dezasamblare in subcomponente a aplicatiilor legacy si reasamblarea automatizata a acestora in medii de tip Cloud computing

Deasemenea in vederea atingerii tuturor elementelor de analiza propuse au fost analizate tehnologii si abordari pentru probleme de inginerie si dezvoltare experimentala cu complexitate stiintifica mai redusa cum ar fi:

1. Modelele de analiza ale tehnologiei de tip baza de date utilizata in aplicatia legacy ce urmeaza a fi migrata
   1. Baze de date flat-file
   2. Baze de date relationale nebazate pe tehnologii client-server
2. Modelele de analiza ale entitatilor de date
   1. Analiza bazata pe parcurgerea si recunoasterea grafurilor
3. Modelele de analiza ale relationarii intre entitatile de date
   1. Analiza bazata pe parcurgerea si recunoasterea grafurilor
4. Modelele de recunoastere a tehnologiei componentelor/modulelor sistemului
   1. Analiza bazata pe citirea componentelor sistemului de operare gazda
5. Modelele de recunoasterele ale layere-lor de comunicare/API/etc.
   1. Analiza bazata pe citirea componentelor sistemului de operare gazda

## Alte remarci

Directorul de proiect intentioneaza ca in baza cercetarii realizate in primul trimestru de implementare a proiectului “Platforma de migrare automatizată în cloud a aplicațiilor și sistemelor informatice clasice Cloudifier.net”, MySMIS: 104349, nr.: P\_38\_543, sa pregateasca si sa publice o lucrare stiintifica intr-o publicatie stiintifica cu recunoastere internationala.

# Rezultatele cercetarii

## Problematica concreta analizata

Scopul analizei stadiului actual al tehnologiei este acela de a determina metodele cele mai moderne/actuale de realizare a predictiilor/inferentelor in imagistica – in particular in cazul proiectului CLOUDIFIER referindu-ne la analiza imaginilor captate in timp real in timpul functionarii aplicatiilor si implicit analiza automatizata cu ajutorul recunoasterii avansate de forme/imagini a aplicatiilor “legacy” in vederea transalatarii acestora automatizate.

In decursul lunii octombrie 2016 au fost analizate cele mai recente si avansate lucrari de cercetare fundamentala si industriala provenite de la cele mai prestigioase institute si universitati printre care enumeram:

* Caltech – California Institute for Technology
* MIT – Massachusetts Institute for Technology
* Stanford
* University of Toronto
* Harvard
* University of Washington

Principalele zone analizate au fost:

* Metodele de tip Deep Learning bazate pe Retele Neuronale Convolutionale – Deep Convolutional Neural Networks
* Metode de tip shallow learning pentru invatarea supervizata a structurilor si a elementelor de imagistica utilizand modele de invatare in timp real (online learning)
* Cele mai moderne abordari in Deep Learning – Tensor Flow
* Cele mai moderne abordari in shallow learning – Extreme Boosted Decision Trees / Random Forests - XGBoost

In decursul lunii noiembrie 2016 au fost analizate cele mai recente si avansate lucrari de cercetare fundamentala in vederea determinarii unui set de algoritmi de Machine Learning ideali pentru identificarea primitivelor de interfata grafica (butoane, campuri, ferestre, texte statice, etc) si a pozitiei acestora in cadrul ecranelor interfetelor grafice

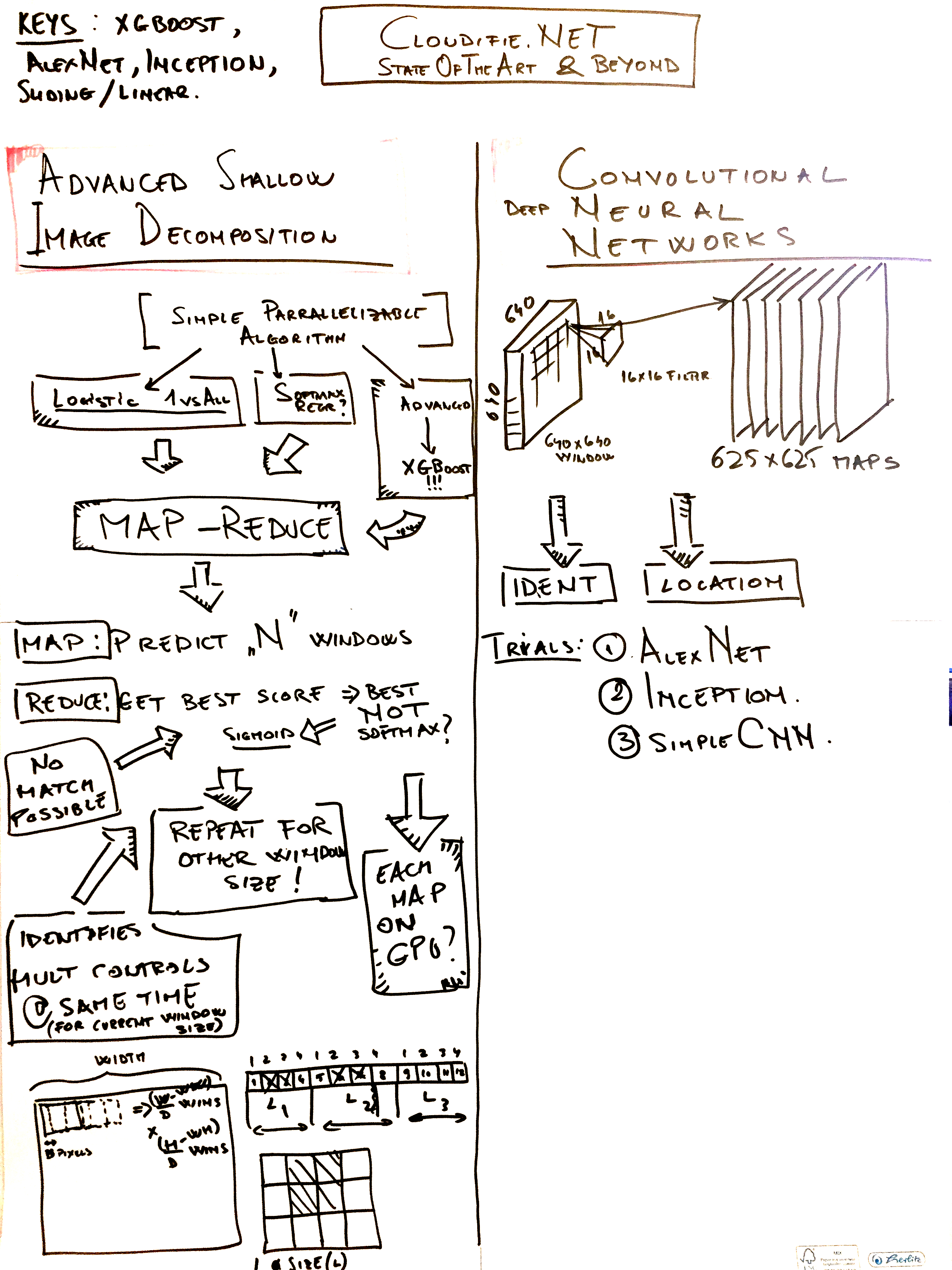
* Determinarea unui algoritm de tip Machine Learning pentru generarea AUTOMATA de interfe grafice si cod sursa aferente pe baza schitelor facute manual pe suport de hartie, tabla, etc
* Analiza TensorFlow
* Analiza XGBoost
* Analiza metode si propuneri pentru biblioteci interne
* Analiza si testarea experimentala a mediilor de procesare numerica masiv paralela cu ajutorul GPU (tehnologiile bazate pe nuclee de calul masiv paralel CUDA)

In decursul luni decembrie 2016 a fost continuat procesul de analiza a stadiului curent al tehnologiei in domeniul sistemelor de tip Machine Learning cu accent pe zona de Deep Learning si in particular a sistemelor de analiza si recunoastere bazata pe inteligenta artificiala a imaginilor. In decursul acestei luni analiza stadiului curent al cercetarii fost fost axat in principal pe lucrarea stiintifica publicata recent de J. Long et al “Fully Convolutional Networks for Semantic Segmentation”, lucrare considerata actualmente state-of-the-art in ceea priveste metodele de recunoastere si segmentare a componentelor in cadrul imaginilor. Pentru referinta prezentam anexat un scurt rezumat in limba engleza a lucrarii de referinta.

Principalele puncte pe care le urmarim in cercetare sunt urmatoarele:

1. Determinarea metodelor optime bazate pe Deep Learning pentru recunoasterea si segmentarea (identificarea locatiei spatiale) a elementelor de interfata grafica pe care Cloudifier.NET va trebuie sa le translateze automatizat din aplicatiile legacy in aplicatiile din mediul cloud computing.
2. Aplicarea de metode simple bazate pe algoritmi de machine learning superficiali (regresie logistica, arbori de decizie, clasificare naiva bazata pe teorema lui Bayes, clusterizare cu analiza distantelor euclidiene) precum si metode de segmentare iterativa a imaginilor analizate cum ar fi metoda ferestrelor deplasate continuu (“ferestre alunecatoare” sau sliding-windows algorithm)

In Imaginea nr. 1 (“Schita de principiu a abordarii stadiului actual al tehnologiei si avansurile aferente”) este prezentata abordarea celor doua metode care vor fi proiectate in paralel in procesul de proiectare a modelelor arhitecturale ale etapei 1.2 din activitatile proiectului si ulterior experimentate in cadrul procesului de dezvoltare experimentala. Detaliile acestei abordari vor fi prezentate in sectiunea “3.3 Doua directii de abordare avansata a cercetarii”

***Imaginea 1 “Schita de principiu a abordarii stadiului actual al tehnologiei si avansurile aferente”***

## Ecosistemul Cloudifier.NET

***Diagrama 2 – Ecosistemul Cloudifier.NET ver 1.0***

|  |
| --- |
| **State-of-the-art** |
| **Shallow Machine Learning User interface analysis:**  **Models for user interface recognition and translation**  **Semantic segmentation and expert systems with DL:**  **Models for data structure analysis and inference**  **Models for stack analysis**  **Models for automated migration** |

|  |  |  |
| --- | --- | --- |
| **Heuristic approach for application migration based on Artificial Intelligence Machine Learning** | | |
| **Cloudifier.NET CAT**  **Automated Translation Engine Architecture** | **Cloudifier.NET Virtual Desktop Renderer** | **Experimental ML Models** |
| **Cloud environment** | | |

Ecosistemul Cloudifier.NET descris in Diagrama 2 cuprinde toate elementele descrise in documentatia Cererii de Finantare si proiectului tehnic aferente proiectului “Platforma de migrare automatizată în cloud a aplicațiilor și sistemelor informatice clasice cloudifier.net” al CLOUDIFIER SRL cu codul MySMIS 2014 nr 104349 si numarul de inregistrare online P\_38\_543. La elementele initiale au fost adaugate si vor fi detaliate pe parcursul activitatilor de cercetare-dezvoltare elementele principale bazate pe tehnologii de Machine Learning.

# Doua directii de abordare avansata a cercetarii

Conform diagramei descrise anterior si a imaginii 1 ce descrie schita de principiu a abordarii proiectului, in urma analizei stadiului actual al tehnologiei au fost determinate doua directii principale in care se vor desfasura activitati de modelare arhitecturala si dezvoltare experiementala in urmatoarea perioada:

1. Modele avansate de recunoastere si decompoziei a cadrelor (imaginilor) din aplicatiile legacy prin utilizarea de algoritmi paralelizabili de predictie bazata pe tehnici de machine learning fara retele neurale adanci
2. Modele de segmentare semantica precum si proiectare de sisteme automatizate expert de tip “Bot” cu ajutorul retelelor neurale adanci convolutionale si recurente (LSTM)

## Tehnici bazate pe modele avansate Shallow Machine Learning

### Paralelizarea masiva a modelelor superfiale de invatare

Prin utilizarea tehnicilor de ultima generatie si in particular a tehnologiei state-of-the-art PASCAL creeata si lansata in 2016 de NVidia se vor utiliza structuri de calcul masiv paralel cu peste 1500 de nuclee de calcul numeric paralelizat.

Aceasta abordare va permite aplicarea de algoritmi de invatare a modelelor superficiale Machine Learning pe structuri de date cu complexitate si dimensionalitate foarte mare – spre exemplu imagini cu adancime de culoare pe 32 de biti (4 octeti) si rezolutii reale de peste W:2000 H:2000. Pentru cazul particular al proiectului Cloudifier.NET este necesara analiza in timp real a imaginilor cu rezolutii inalte peste 1080x786x4 ceea ce genereaza un minim de 829,000 de dimensiuni (variabile predictor) . Astfel se vor putea aplica algoritmi de tip regresie logistica softmax antrenata prin mini-calupuri informationale pe structura de calcul masiv paralel.

Detaliile modului de aplicare a algoritmilor vor fi definite in etapa dezvoltarii subactivitatii 1.2 de proiectare a modelelor arhitecturale

Print utilizarea celor mai recente cercetarii in domeniul Machine Learning pentru ansamble de modele se urmareste testarea experimentala a potentialei utilizari de modele simple si eficiente pentru toate componentele de inteligenta artificiala ale proiectului Cloudifier.NET. Aceste componente vor fi asociate cu algoritmi clasici de tip “Sliding Windows” – utilizati actualmente de camerele de luat vederi de ultima generatie - care vor genera mecanic segmentele ce urmeaza sa fie analizate de algoritmii de tip Machine Learning.

In cele ce urmeaza este prezentat un test realizat cu ajutorul tehnologiei CUDA a NVidia pentru calcul masiv paralel utilizand limbajul de nivel inalt Python

|  |
| --- |
| [1](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_1) # Exercise 1 from http://webapp.dam.brown.edu/wiki/SciComp/CudaExercises  [2](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_2)  [3](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_3) # Transposition of a matrix  [4](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_4) # by Hendrik Riedmann <riedmann@dam.brown.edu>  [5](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_5)  [6](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_6) from \_\_future\_\_ import division, print\_function  [7](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_7)  [8](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_8) import pycuda.driver as cuda  [9](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_9) import pycuda.gpuarray as gpuarray  [10](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_10) import pycuda.autoinit  [11](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_11) from pycuda.compiler import SourceModule  [12](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_12)  [13](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_13) import numpy  [14](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_14) import numpy.linalg as la  [15](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_15)  [16](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_16) from pycuda.tools import context\_dependent\_memoize  [17](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_17)  [18](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_18) block\_size = 16  [19](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_19)  [20](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_20) @context\_dependent\_memoize  [21](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_21) def \_get\_transpose\_kernel():  [22](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_22) mod = SourceModule("""  [23](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_23) #define BLOCK\_SIZE %(block\_size)d  [24](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_24) #define A\_BLOCK\_STRIDE (BLOCK\_SIZE \* a\_width)  [25](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_25) #define A\_T\_BLOCK\_STRIDE (BLOCK\_SIZE \* a\_height)  [26](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_26)  [27](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_27) \_\_global\_\_ void transpose(float \*A\_t, float \*A, int a\_width, int a\_height)  [28](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_28) {  [29](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_29) // Base indices in A and A\_t  [30](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_30) int base\_idx\_a = blockIdx.x \* BLOCK\_SIZE +  [31](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_31) blockIdx.y \* A\_BLOCK\_STRIDE;  [32](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_32) int base\_idx\_a\_t = blockIdx.y \* BLOCK\_SIZE +  [33](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_33) blockIdx.x \* A\_T\_BLOCK\_STRIDE;  [34](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_34)  [35](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_35) // Global indices in A and A\_t  [36](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_36) int glob\_idx\_a = base\_idx\_a + threadIdx.x + a\_width \* threadIdx.y;  [37](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_37) int glob\_idx\_a\_t = base\_idx\_a\_t + threadIdx.x + a\_height \* threadIdx.y;  [38](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_38)  [39](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_39) \_\_shared\_\_ float A\_shared[BLOCK\_SIZE][BLOCK\_SIZE+1];  [40](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_40)  [41](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_41) // Store transposed submatrix to shared memory  [42](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_42) A\_shared[threadIdx.y][threadIdx.x] = A[glob\_idx\_a];  [43](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_43)  [44](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_44) \_\_syncthreads();  [45](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_45)  [46](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_46) // Write transposed submatrix to global memory  [47](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_47) A\_t[glob\_idx\_a\_t] = A\_shared[threadIdx.x][threadIdx.y];  [48](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_48) }  [49](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_49) """% {"block\_size": block\_size})  [50](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_50)  [51](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_51) func = mod.get\_function("transpose")  [52](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_52) func.prepare("PPii")  [53](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_53)  [54](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_54) from pytools import Record  [55](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_55) class TransposeKernelInfo(Record): pass  [56](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_56)  [57](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_57) return TransposeKernelInfo(func=func,  [58](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_58) block=(block\_size, block\_size, 1),  [59](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_59) block\_size=block\_size,  [60](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_60) granularity=block\_size)  [61](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_61)  [62](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_62)  [63](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_63)  [64](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_64) def \_get\_big\_block\_transpose\_kernel():  [65](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_65) mod = SourceModule("""  [66](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_66) #define BLOCK\_SIZE %(block\_size)d  [67](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_67) #define A\_BLOCK\_STRIDE (BLOCK\_SIZE \* a\_width)  [68](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_68) #define A\_T\_BLOCK\_STRIDE (BLOCK\_SIZE \* a\_height)  [69](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_69)  [70](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_70) \_\_global\_\_ void transpose(float \*A, float \*A\_t, int a\_width, int a\_height)  [71](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_71) {  [72](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_72) // Base indices in A and A\_t  [73](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_73) int base\_idx\_a = 2 \* blockIdx.x \* BLOCK\_SIZE +  [74](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_74) 2 \* blockIdx.y \* A\_BLOCK\_STRIDE;  [75](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_75) int base\_idx\_a\_t = 2 \* blockIdx.y \* BLOCK\_SIZE +  [76](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_76) 2 \* blockIdx.x \* A\_T\_BLOCK\_STRIDE;  [77](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_77)  [78](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_78) // Global indices in A and A\_t  [79](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_79) int glob\_idx\_a = base\_idx\_a + threadIdx.x + a\_width \* threadIdx.y;  [80](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_80) int glob\_idx\_a\_t = base\_idx\_a\_t + threadIdx.x + a\_height \* threadIdx.y;  [81](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_81)  [82](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_82) \_\_shared\_\_ float A\_shared[2 \* BLOCK\_SIZE][2 \* BLOCK\_SIZE + 1];  [83](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_83)  [84](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_84) // Store transposed submatrix to shared memory  [85](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_85) A\_shared[threadIdx.y][threadIdx.x] = A[glob\_idx\_a];  [86](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_86) A\_shared[threadIdx.y][threadIdx.x + BLOCK\_SIZE] =  [87](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_87) A[glob\_idx\_a + A\_BLOCK\_STRIDE];  [88](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_88) A\_shared[threadIdx.y + BLOCK\_SIZE][threadIdx.x] =  [89](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_89) A[glob\_idx\_a + BLOCK\_SIZE];  [90](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_90) A\_shared[threadIdx.y + BLOCK\_SIZE][threadIdx.x + BLOCK\_SIZE] =  [91](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_91) A[glob\_idx\_a + BLOCK\_SIZE + A\_BLOCK\_STRIDE];  [92](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_92)  [93](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_93) \_\_syncthreads();  [94](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_94)  [95](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_95) // Write transposed submatrix to global memory  [96](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_96) A\_t[glob\_idx\_a\_t] = A\_shared[threadIdx.x][threadIdx.y];  [97](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_97) A\_t[glob\_idx\_a\_t + A\_T\_BLOCK\_STRIDE] =  [98](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_98) A\_shared[threadIdx.x + BLOCK\_SIZE][threadIdx.y];  [99](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_99) A\_t[glob\_idx\_a\_t + BLOCK\_SIZE] =  [100](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_100) A\_shared[threadIdx.x][threadIdx.y + BLOCK\_SIZE];  [101](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_101) A\_t[glob\_idx\_a\_t + A\_T\_BLOCK\_STRIDE + BLOCK\_SIZE] =  [102](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_102) A\_shared[threadIdx.x + BLOCK\_SIZE][threadIdx.y + BLOCK\_SIZE];  [103](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_103) }  [104](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_104) """% {"block\_size": block\_size})  [105](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_105)  [106](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_106) func = mod.get\_function("transpose")  [107](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_107) func.prepare("PPii")  [108](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_108)  [109](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_109) from pytools import Record  [110](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_110) class TransposeKernelInfo(Record): pass  [111](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_111)  [112](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_112) return TransposeKernelInfo(func=func,  [113](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_113) block=(block\_size, block\_size, 1),  [114](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_114) block\_size=block\_size,  [115](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_115) granularity=2\*block\_size)  [116](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_116)  [117](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_117)  [118](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_118)  [119](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_119)  [120](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_120) def \_transpose(tgt, src):  [121](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_121) krnl = \_get\_transpose\_kernel()  [122](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_122)  [123](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_123) w, h = src.shape  [124](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_124) assert tgt.shape == (h, w)  [125](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_125) assert w % krnl.granularity == 0  [126](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_126) assert h % krnl.granularity == 0  [127](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_127)  [128](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_128) krnl.func.prepared\_call(  [129](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_129) (w // krnl.granularity, h // krnl.granularity), krnl.block,  [130](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_130) tgt.gpudata, src.gpudata, w, h)  [131](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_131)  [132](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_132)  [133](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_133)  [134](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_134)  [135](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_135) def transpose(src):  [136](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_136) w, h = src.shape  [137](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_137)  [138](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_138) result = gpuarray.empty((h, w), dtype=src.dtype)  [139](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_139) \_transpose(result, src)  [140](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_140) return result  [141](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_141)  [142](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_142)  [143](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_143)  [144](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_144)  [145](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_145)  [146](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_146) def check\_transpose():  [147](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_147) from pycuda.curandom import rand  [148](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_148)  [149](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_149) for i in numpy.arange(10, 13, 0.125):  [150](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_150) size = int(((2\*\*i) // 32) \* 32)  [151](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_151) print(size)  [152](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_152)  [153](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_153) source = rand((size, size), dtype=numpy.float32)  [154](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_154)  [155](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_155) result = transpose(source)  [156](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_156)  [157](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_157) err = source.get().T - result.get()  [158](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_158) err\_norm = la.norm(err)  [159](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_159)  [160](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_160) source.gpudata.free()  [161](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_161) result.gpudata.free()  [162](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_162)  [163](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_163) assert err\_norm == 0, (size, err\_norm)  [164](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_164)  [165](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_165)  [166](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_166)  [167](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_167)  [168](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_168) def run\_benchmark():  [169](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_169) from pycuda.curandom import rand  [170](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_170)  [171](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_171) powers = numpy.arange(10, 13, 2\*\*(-6))  [172](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_172) sizes = [int(size) for size in numpy.unique(2\*\*powers // 16 \* 16)]  [173](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_173) bandwidths = []  [174](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_174) times = []  [175](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_175)  [176](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_176) for size in sizes:  [177](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_177)  [178](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_178) source = rand((size, size), dtype=numpy.float32)  [179](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_179) target = gpuarray.empty((size, size), dtype=source.dtype)  [180](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_180)  [181](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_181) start = pycuda.driver.Event()  [182](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_182) stop = pycuda.driver.Event()  [183](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_183)  [184](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_184) warmup = 2  [185](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_185)  [186](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_186) for i in range(warmup):  [187](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_187) \_transpose(target, source)  [188](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_188)  [189](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_189) count = 10  [190](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_190)  [191](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_191) cuda.Context.synchronize()  [192](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_192) start.record()  [193](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_193)  [194](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_194) for i in range(count):  [195](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_195) \_transpose(target, source)  [196](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_196)  [197](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_197) stop.record()  [198](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_198) stop.synchronize()  [199](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_199)  [200](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_200) elapsed\_seconds = stop.time\_since(start)\*1e-3  [201](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_201) mem\_bw = source.nbytes / elapsed\_seconds \* 2 \* count  [202](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_202)  [203](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_203) bandwidths.append(mem\_bw)  [204](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_204) times.append(elapsed\_seconds)  [205](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_205)  [206](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_206) slow\_sizes = [s for s, bw in zip(sizes, bandwidths) if bw < 40e9]  [207](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_207) print("Sizes for which bandwidth was low:", slow\_sizes)  [208](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_208) print("Ditto, mod 64:", [s % 64 for s in slow\_sizes])  [209](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_209) from matplotlib.pyplot import semilogx, loglog, show, savefig, clf, xlabel, ylabel  [210](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_210) xlabel('matrix size')  [211](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_211) ylabel('bandwidth')  [212](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_212) semilogx(sizes, bandwidths)  [213](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_213) savefig("transpose-bw.png")  [214](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_214) clf()  [215](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_215) xlabel('matrix size')  [216](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_216) ylabel('time')  [217](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_217) loglog(sizes, times)  [218](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_218) savefig("transpose-times.png")  [219](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_219)  [220](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_220)  [221](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_221)  [222](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_222)  [223](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_223) #check\_transpose()  [224](https://wiki.tiker.net/PyCuda/Examples/MatrixTranspose#CA-10556369a54a4697c4ad803a68bb827fd80493f5_224) run\_benchmark() |

### Framework-ul state-of-the-art XGBoost

In vederea aplicarii celor mai recente cercetari in domeniile vizate a fost selectat printre modelele state-of-the-art si framework-ul XGBoost dezvoltat in perioada 2015-2016, framework ce a fost popularizat in perioada desfasurata dupa depunerea aplicatiei initiale a proiectului Cloudifier si actualmente este considerat state-of-the-art in domeniul modelelor de tip ansamblu de algoritmi de inteligenta artificala destinati predictiei.

In continuare prezentam testele realizate pe aceasta tehnologie pe parcursul desfasurarii activitatilor 1.1 – Analiza State-of-the-Art

|  |
| --- |
| """  Created on: 10/27/2016  Last Modified: 12/23/2016  @author: Andrei Ionut DAMIAN  """  # Tune learning\_rate  from numpy import loadtxt  from xgboost import XGBClassifier  from sklearn.grid\_search import GridSearchCV  from sklearn.cross\_validation import StratifiedKFold  from sklearn.ensemble import RandomForestClassifier  from sklearn.ensemble import VotingClassifier  from sklearn.linear\_model import LogisticRegression  from sklearn.svm import SVC  from sklearn.neighbors import KNeighborsClassifier  from sklearn.model\_selection import GridSearchCV  from xgboost import XGBClassifier  from xgboost import XGBRegressor  import pandas as pd  import numpy as np  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score  from sklearn.metrics import mean\_absolute\_error  from sklearn.metrics import make\_scorer  from sklearn.preprocessing import StandardScaler  import sys  import re  import time  def copy\_from\_combined(train\_ds, test\_ds, combined\_ds, columns, index\_pos):  for column in columns:  train\_ds.set\_value(train\_ds.index, column, combined\_ds[:index\_pos][column].values)  test\_ds.set\_value(test\_ds.index, column, combined\_ds[index\_pos:][column].values)  if \_\_name\_\_ == "\_\_main\_\_":    Verbose = False    init\_train\_data = pd.read\_csv('train.csv')  init\_test\_data = pd.read\_csv('test.csv')      train\_data = init\_train\_data.copy()  test\_data = init\_test\_data.copy()  sex\_values={'male':1,'female':0}    train\_data['Sex']=train\_data['Sex'].replace(sex\_values)    test\_data['Sex']=test\_data['Sex'].replace(sex\_values)    train\_data.set\_value(train\_data.Embarked.isnull(), "Embarked", "C")  test\_data.set\_value(test\_data.Fare.isnull(), 'Fare', 8.05) # the 60 year old guy    train\_data.set\_value(train\_data.Cabin.isnull(), 'Cabin', 'U0')  test\_data.set\_value(test\_data.Cabin.isnull(), 'Cabin', 'U0')  ###  ### feature engineering  ### all features will begin with EF\_ prefix  ###  # create nr of names feature  train\_names = train\_data.Name.map(lambda x: len(re.split(' ', x)))  train\_data.set\_value(train\_data.index, 'EF\_Names', train\_names)  del train\_names  test\_names = test\_data.Name.map(lambda x: len(re.split(' ', x)))  test\_data.set\_value(test\_data.index, 'EF\_Names', test\_names)  del test\_names    #create title feature (later will be factorized/O.H.E.-ed)  prog = re.compile(', (.\*?)\.')    title = train\_data.Name.map(lambda x: prog.findall(x)[0])  title[title=='Mme'] = 'Mrs'  title[title.isin(['Ms','Mlle'])] = 'Miss'  title[title.isin(['Don', 'Jonkheer'])] = 'Sir'  title[title.isin(['Dona', 'Lady', 'the Countess'])] = 'Lady'  title[title.isin(['Capt', 'Col', 'Major', 'Dr', 'Officer', 'Rev'])] = 'Officer'  \_ = train\_data.set\_value(train\_data.index, 'EF\_Title', title)  del title  title = test\_data.Name.map(lambda x: prog.findall(x)[0])  title[title=='Mme'] = 'Mrs'  title[title.isin(['Ms','Mlle'])] = 'Miss'  title[title.isin(['Don', 'Jonkheer'])] = 'Sir'  title[title.isin(['Dona', 'Lady', 'the Countess'])] = 'Lady'  title[title.isin(['Capt', 'Col', 'Major', 'Dr', 'Officer', 'Rev'])] = 'Officer'  \_ = test\_data.set\_value(test\_data.index, 'EF\_Title', title)  del title      ##  ## from now on we will combine both TRAIN and TEST in order to computer  ## accurate averages/predictions for certain variables and engineered  ## predictors that we will construct  ##  combined = pd.concat([train\_data, test\_data], ignore\_index=True)  ## Generate "Deck" feature - factorization of Cabin 1st letter  deck = combined[~combined.Cabin.isnull()].Cabin.map( lambda x : re.compile("([a-zA-Z]+)").search(x).group())  deck = pd.factorize(deck)[0]  combined.set\_value(combined.index, 'EF\_Deck', deck)  del deck  ## Extract "Room" feature - the number contained in the cabin  ## info. Standardize the result  checker = re.compile("([0-9]+)")  def roomNum(x):  nums = checker.search(x)  if nums:  return int(nums.group())+1  else:  return 1  rooms = combined.Cabin.map(roomNum)  \_ = combined.set\_value(combined.index, 'EF\_Room', rooms)  combined['EF\_Room'] = (combined.EF\_Room-combined.EF\_Room.min())/combined.EF\_Room.max()  del checker, roomNum  ## now compute Group size  combined['EF\_Group\_size'] = combined.Parch + combined.SibSp + 1  combined['EF\_Group\_type'] = pd.Series('M', index=combined.index)  combined.set\_value(combined.EF\_Group\_size >4, 'EF\_Group\_type', 'L')  combined.set\_value(combined.EF\_Group\_size==1, 'EF\_Group\_type', 'S')  ##  # save work in train/test  saved\_cols = ['EF\_Deck', 'EF\_Room', 'EF\_Group\_type', 'EF\_Group\_size',  'EF\_Names', 'EF\_Title']  copy\_from\_combined(train\_data, test\_data, combined, saved\_cols, 891 )  # done saving    # transform 'Embarked', 'Sex', 'EF\_Title', 'EF\_Group\_type' in OH features  # normalize Fare  scaler = StandardScaler()  combined['EF\_Fare'] = pd.Series(scaler.fit\_transform(combined.Fare.reshape(-1,1)).reshape(-1),  index=combined.index)  ## now lets drop some of standard predictors that have been  ## re-engineered    combined.drop(labels=['PassengerId', 'Name', 'Cabin', 'Survived', 'Ticket', 'Fare'],  axis=1, inplace=True)  ## now lets encode as planned  combined = pd.get\_dummies(combined,  columns=['Embarked', 'Sex', 'EF\_Title', 'EF\_Group\_type'])  # now FINALLY predict AGE  X\_train\_age = combined[~combined.Age.isnull()].drop('Age', axis=1)  y\_train\_age = combined[~combined.Age.isnull()].Age  xgb\_regressor = XGBRegressor(max\_depth=4)  regr\_scoring = make\_scorer(mean\_absolute\_error, greater\_is\_better=False)  regr\_parameters = {'reg\_alpha':np.linspace(0.1,1.0,5),  'reg\_lambda': np.linspace(1.0,3.0,5)}    tmp\_rgr = GridSearchCV(xgb\_regressor,  param\_grid = regr\_parameters,  scoring = regr\_scoring,  n\_jobs=6)    tmp\_rgr.fit(X\_train\_age, y\_train\_age)  reg\_xgb = tmp\_rgr.best\_estimator\_  age\_preds = reg\_xgb.predict(combined[combined.Age.isnull()].drop('Age', axis=1))  combined.set\_value(combined.Age.isnull(), 'Age', age\_preds)  combined['EF\_NorAge'] = pd.Series(scaler.fit\_transform(combined.Age.reshape(-1,1)).reshape(-1),  index=combined.index)  combined['EF\_NorNames'] = pd.Series(scaler.fit\_transform(combined.EF\_Names.reshape(-1,1)).reshape(-1),  index=combined.index)  combined['EF\_Group\_size'] = pd.Series(scaler.fit\_transform(combined.EF\_Group\_size.reshape(-1,1)).reshape(-1),  index=combined.index)  # now save processed engineered features  saved\_cols = ['EF\_NorAge', 'EF\_NorNames', 'EF\_Group\_size', 'EF\_Fare']  copy\_from\_combined(train\_data, test\_data, combined, saved\_cols, 891 )  # done  # clean features that have been transformed  train\_data.drop(labels=['PassengerId', 'Name', 'EF\_Names', 'Cabin', 'Ticket', 'Age', 'Fare'],  axis=1, inplace=True)  test\_data.drop(labels= ['Name', 'EF\_Names', 'Cabin', 'Ticket', 'Age', 'Fare'],  axis=1, inplace=True)  train\_data = pd.get\_dummies(train\_data, columns=['Embarked', 'Pclass', 'EF\_Title', 'EF\_Group\_type'])  test\_data = pd.get\_dummies(test\_data, columns=['Embarked', 'Pclass', 'EF\_Title', 'EF\_Group\_type'])  test\_data['EF\_Title\_Sir'] = pd.Series(0, index=test\_data.index) # missing in test dataset      X = train\_data.drop(['Survived'], axis=1)  Y = train\_data.Survived    # grid search  model = XGBClassifier()  learning\_rate = [0.01, 0.1, 0.2, 0.3]  reg\_lambda = [ 2.0, 3.0, 4.0, 5.0]  reg\_alpha =[2.0, 3.0, 4.0, 5.0]  n\_estimators = [100, 400]  max\_depth = [4, 6, 8]  param\_grid = dict(max\_depth = max\_depth,  n\_estimators = n\_estimators,  learning\_rate = learning\_rate,  reg\_alpha = reg\_alpha,  reg\_lambda = reg\_lambda)  kfold = StratifiedKFold(Y, n\_folds=10, shuffle=True)  scoring = make\_scorer(accuracy\_score, greater\_is\_better=True)  grid\_search = GridSearchCV(model,  param\_grid,  scoring= scoring,  n\_jobs=-1,  cv=kfold)  result = grid\_search.fit(X, Y)  final\_clf = result.best\_estimator\_  # summarize results  means, stdevs = [], []  for params, mean\_score, scores in result.grid\_scores\_:  stdev = scores.std()  means.append(mean\_score)  stdevs.append(stdev)  print("%f (%f) with: %r" % (mean\_score, stdev, params))  print("Best: %f using %s" % (result.best\_score\_, result.best\_params\_))  SAVE = True  if SAVE:  PassengerId = test\_data.PassengerId  test\_data.drop("PassengerId", axis = 1, inplace = True)  test\_data = test\_data[X.columns]  y\_f\_preds = final\_clf.predict(test\_data)  save\_data = pd.DataFrame(columns = ['PassengerId', 'Survived'])  save\_data.PassengerId = PassengerId  save\_data.Survived = pd.Series(y\_f\_preds, index=save\_data.index)    file\_name = str(time.time())+'\_pxgb\_predictions.csv'  save\_data.to\_csv(file\_name, index=False) |

### Modele proprii testate in Eigen

O alta abordare in vederea determinarii celei mai eficiente si fiabile solutii pentru implementarea componentelor de Machine Learning din cadrul Cloudifier.NET este utilizarea bibliotecilor de nivel scazut scrise in C++, biblioteci ce formeaza actualmente state-of-the-art in aceasta zona si au fost analizate in cadrul etapei/subactivitatii 1.1 “Analiza state-of-the-art” a proiectului. In particular au fost analizate bibliotecile EIGNEN (<http://eigen.tuxfamily.org/index.php?title=Main_Page> sau <https://en.wikipedia.org/wiki/Eigen_(C%2B%2B_library)> )

In continuare prezentam testele realizate pe aceasta tehnologie pe parcursul desfasurarii activitatilor 1.1 – Analiza State-of-the-Art. Aceste teste reprezinta implementarea la nivelul cel mai jos al nucleului de calcul a algoritmului de Machine Learning bazat regresia logistice softmax.

|  |
| --- |
| //  // EigenEngine - ML engine based on Eigen library  //  // Created: 11/01/2016  // Last modified: 12/29/2016  //  // @Author: Andrei Ionut DAMIAN  // @Contributor: Octavian BULIE  #pragma once  #include "stdafx.h"  #include "stdio.h"  #include <string>  #include <iostream>  #include <fstream>  #include <vector>  #include <set>  #include <Eigen/Dense>  #include <Eigen/Core>  #include <Eigen/SVD>  #include <sys/stat.h>  #include <chrono>  #include <algorithm>  #include <random>  using namespace std;  using namespace std::chrono;  using Eigen::MatrixXd;  using Eigen::VectorXd;  using namespace Eigen;  using namespace std;  struct TrainCrossSplits  {  MatrixXd X\_train;  MatrixXd X\_cross;  vector <string> Labels;  VectorXd y\_train;  VectorXd y\_cross;  };  class GenericEngine  {  private:  long LoadedDataNrFields;  long LoadedDataNrRows;  long TrainTestSplitPos;  std::default\_random\_engine random\_engine;  protected:  milliseconds start\_time;  milliseconds end\_time;  bool bBiasAdded; // variable that stores bias information for pre-loaded data  string CLF\_NAME;  long NR\_FEATS;  long NR\_CLASSES;  public:  bool VERBOSE\_ENGINE;  MatrixXd \*X\_loaded;  VectorXd \*y\_loaded;  MatrixXd \*X\_train;  VectorXd \*y\_train;  MatrixXd \*X\_cross;  VectorXd \*y\_cross;  MatrixXd \*LoadedData;  vector <string> LoadedDataHeader;  vector <string> LabelsVector;  GenericEngine()  {  CLF\_NAME = "Generic Engine";  // obtain a time-based seed:  unsigned seed = std::chrono::system\_clock::now().time\_since\_epoch().count();  random\_engine = default\_random\_engine(seed);  VERBOSE\_ENGINE = true;  bBiasAdded = false;  X\_train = NULL;  y\_train = NULL;  X\_loaded = NULL;  y\_loaded = NULL;  X\_cross = NULL;  y\_cross = NULL;  LoadedData = NULL;  }  bool file\_exists(const std::string& name);  void debug\_info(string str\_message)  {  if (VERBOSE\_ENGINE)  printf("\n[DEBUG] %s", str\_message.c\_str());  }  void debug\_info(string msg, MatrixXd mat)  {  std::stringstream ss;  ss << mat;  string str\_matrix = ss.str();  string msgp = "\n[DEBUG] " + msg + "\n";  printf(msgp.c\_str());  std::cout << str\_matrix << std::endl;  }  void debug\_info(MatrixXd mat)  {  std::stringstream ss;  ss << mat;  string str\_matrix = ss.str();  printf("\n[DEBUG] Matrix:\n");  std::cout << str\_matrix << std::endl;  }  void debug\_info(VectorXd vec, bool bHorizontal)  {  std::stringstream ss;  if (bHorizontal)  ss << vec.transpose();  else  ss << vec;  string str\_vector = ss.str();  printf("\n[DEBUG] Vector:\n");  std::cout << str\_vector << std::endl;  }  void debug\_info(string msg, VectorXd vec, bool bHorizontal)  {  std::stringstream ss;  if (bHorizontal)  ss << vec.transpose();  else  ss << vec;  string str\_vector = ss.str();  string msgp = "\n[DEBUG] " + msg + "\n";  printf(msgp.c\_str());  std::cout << str\_vector << std::endl;  }  void debug\_info()  {  printf("\n[DEBUG] [PRESS ENTER]");  int c = getc(stdin);  }  MatrixXd ShuffleMatrixRows(MatrixXd DataMatrix);  int FindLabelId(vector <string> labels, string value);  void BeginTimer();  long EndTimer();  vector <string> ToLabels(VectorXd y);  TrainCrossSplits LoadCSV(const string& inputfile, const bool bShuffle = false, const bool bAddBias = false);  };  class GenericLinearEngine : public GenericEngine  {  protected:  long nr\_batches; // how many batches have been processed (epochs, online trainings, etc)  VectorXd \*SingleClassTheta;  MatrixXd \*Theta;  VectorXd \*J\_values;  void add\_cost(double J);  private:  void init();  public:  GenericLinearEngine()  {  CLF\_NAME = "VIRTUAL Generic Linear Engine";  init();  }  ~GenericLinearEngine()  {  debug\_info("Deleting object [" + CLF\_NAME + "]");  if (LoadedData != NULL)  delete LoadedData;  if (X\_loaded != NULL)  delete X\_loaded;  if (y\_loaded != NULL)  delete y\_loaded;  if (X\_train != NULL)  delete X\_train;  if (y\_train != NULL)  delete y\_train;  if (X\_cross != NULL)  delete X\_cross;  if (y\_cross != NULL)  delete y\_cross;  if (Theta != NULL)  delete Theta;  if (SingleClassTheta != NULL)  delete SingleClassTheta;  if (J\_values != NULL)  delete J\_values;  }  VectorXd PredictSingleClass(MatrixXd X);  virtual MatrixXd Predict(MatrixXd X);  vector <string> PredictLabels(MatrixXd X);  vector <string> PredictLabelsUsingYHat(MatrixXd y\_hat);  string GetName();  MatrixXd& GetTheta();  float NRMSE(VectorXd y\_hat, VectorXd y);  float RMSE(VectorXd y\_hat, VectorXd y);  float CrossEvaluationSingleClass(bool bClass);  float TrainEvaluationSingleClass(bool bClass);  float CrossEvaluation(bool bClass);  float TrainEvaluation(bool bClass);  };  class NormalRegressor : public GenericLinearEngine  {  protected:    int t;  public:  NormalRegressor()  {  NR\_FEATS = 0;  NR\_CLASSES = 0;  CLF\_NAME = "Batch Normal Regressor";  }  void Train(MatrixXd X, MatrixXd y);  void Train();  };  class OnlineClassifier : public GenericLinearEngine  {  protected:  // temp variables  MatrixXd LastYHat;  MatrixXd LastGrad;  MatrixXd LastXObs;  MatrixXd LastYOHM;  MatrixXd LastYERR;  double LearningRate;  MatrixXd softmax(MatrixXd z);  double cross\_entropy(MatrixXd yOHM, MatrixXd y\_hat);  public:  OnlineClassifier(int nr\_features, int nr\_classes, vector <string> &labels, double alpha\_learning\_rate)  {  CLF\_NAME = "Online Linear Classifier";  NR\_FEATS = nr\_features;  NR\_CLASSES = nr\_classes;  LabelsVector = labels;  LearningRate = alpha\_learning\_rate;  Theta = new MatrixXd(NR\_FEATS+1, NR\_CLASSES); // add 1 row for biases  Theta->fill(0);  }  void SimulateOnlineTrain();  void OnlineTrain(MatrixXd xi, VectorXd yi);  double CostFunction();  MatrixXd Predict(MatrixXd X);  };  //  // BEGIN Generic Engine Class definitions - basic ancestor helper class  //  inline bool GenericEngine::file\_exists(const std::string & name)  {  if (FILE \*file = fopen(name.c\_str(), "r")) {  fclose(file);  return true;  }  else {  return false;  }  }  inline MatrixXd GenericEngine::ShuffleMatrixRows(MatrixXd DataMatrix)  {  long size = DataMatrix.rows();  PermutationMatrix<Dynamic, Dynamic> perm(size);  perm.setIdentity();  std::shuffle(perm.indices().data(),  perm.indices().data() + perm.indices().size(),  this->random\_engine);  MatrixXd A\_perm = perm \* DataMatrix; // permute rows  return(A\_perm);  }  inline int GenericEngine::FindLabelId(vector<string> labels, string value)  {  int pos = find(labels.begin(), labels.end(), value) - labels.begin();  if (pos >= labels.size()) {  //old\_name\_ not found  pos = -1;  }  return(pos);  }  void GenericEngine::BeginTimer()  {  milliseconds ms = duration\_cast< milliseconds >(  system\_clock::now().time\_since\_epoch()  );  start\_time = ms;  }  inline long GenericEngine::EndTimer()  {  milliseconds ms = duration\_cast< milliseconds >(  system\_clock::now().time\_since\_epoch()  );  end\_time = ms;  return (end\_time - start\_time).count();  }  inline vector<string> GenericEngine::ToLabels(VectorXd y)  {  vector <string> labels;  for (long i = 0;i < y.size();i++)  {  string s = LabelsVector[y(i)];  labels.push\_back(s);  }  return(labels);  }  inline TrainCrossSplits GenericEngine::LoadCSV(const string & inputfile, const bool bShuffle, const bool bAddBias)  {  int nr\_rows = 0;  int nr\_cols = 0;  string fname = inputfile;  TrainCrossSplits rec\_results;  if (!file\_exists(inputfile))  throw std::invalid\_argument("Received invalid file in LoadCSV: " + fname);  ifstream infile(fname, std::ifstream::in);  if (!infile.good())  throw std::invalid\_argument("Received invalid file in LoadCSV: " + fname);  debug\_info("Loading " + fname + " dataset...");  vector< vector<string> > result;  while (!infile.eof())  {  //go through every line  string line;  getline(infile, line);  vector <string> record;  nr\_cols = 0;  std::size\_t prev = 0, pos;  while ((pos = line.find\_first\_of(",;", prev)) != std::string::npos)  {  if (pos > prev)  {  record.push\_back(line.substr(prev, pos - prev));  nr\_cols++;  }  prev = pos + 1;  }  if (prev < line.length())  {  record.push\_back(line.substr(prev, std::string::npos));  nr\_cols++;  }  if (nr\_cols > 0)  {  result.push\_back(record);  nr\_rows++;  }  }  //  // now load whole data, X and y matrices  // assume last column of loaded data is the results / labels  //  LoadedDataNrFields = result[0].size();  LoadedDataNrRows = nr\_rows - 1; // rows minus field names row  debug\_info("Loaded " + std::to\_string(LoadedDataNrRows) + " X " + std::to\_string(LoadedDataNrFields) + " dataset");  LoadedData = new MatrixXd(LoadedDataNrRows, LoadedDataNrFields);  y\_loaded = new VectorXd(LoadedDataNrRows);  X\_loaded = new MatrixXd(LoadedDataNrRows, LoadedDataNrFields - 1);  std::set <string> LabelsSet;  long i, j;  for (j = 0;j < LoadedDataNrFields;j++)  LoadedDataHeader.push\_back((string)result[0][j]);  //  // assume dataset is curated and ONLY last column contains text labels  //  vector <string> loaded\_labels;  for (i = 0;i < LoadedDataNrRows;i++)  for (j = 0;j < LoadedDataNrFields;j++)  {  double fcell = 0;  string scell = result[i + 1][j];  try  {  if (j != ((LoadedDataNrFields - 1)))  fcell = ::atof(scell.c\_str());  }  catch (...)  {  }  (\*LoadedData)(i, j) = fcell;  if (j == (LoadedDataNrFields - 1))  {  LabelsSet.insert(scell);  loaded\_labels.push\_back(scell);  }  }  LabelsVector.assign(LabelsSet.begin(), LabelsSet.end());  for (int label\_idx = 0;label\_idx < loaded\_labels.size();label\_idx++)  {  string c\_label = loaded\_labels[label\_idx];  int iLabel = FindLabelId(LabelsVector, c\_label);  (\*LoadedData)(label\_idx, LoadedDataNrFields - 1) = iLabel;  }  if (bShuffle)  {  MatrixXd ttt = ShuffleMatrixRows(\*LoadedData);  \*LoadedData = ttt;  }  float test\_size = 0.2;  int test\_rows = LoadedDataNrRows \* test\_size;  int train\_rows = LoadedDataNrRows - test\_rows;  TrainTestSplitPos = train\_rows;  \*X\_loaded = LoadedData->leftCols(LoadedDataNrFields - 1);  \*y\_loaded = LoadedData->rightCols(1);  NR\_FEATS = X\_loaded->cols();  NR\_CLASSES = LabelsVector.size();  if (bAddBias)  {  // now add bias  VectorXd bias(LoadedDataNrRows);  bias.fill(1);  MatrixXd \*TempX = new MatrixXd(LoadedDataNrRows, LoadedDataNrFields - 1 + 1); // bias size  \*TempX << bias, \*X\_loaded;  bBiasAdded = true;  delete X\_loaded;  X\_loaded = TempX;  // done adding bias  }  X\_train = new MatrixXd(X\_loaded->topRows(train\_rows));  X\_cross = new MatrixXd(X\_loaded->bottomRows(test\_rows));  y\_train = new VectorXd(y\_loaded->head(train\_rows));  y\_cross = new VectorXd(y\_loaded->tail(test\_rows));  rec\_results.X\_cross = \*X\_cross;  rec\_results.X\_train = \*X\_train;  rec\_results.y\_cross = \*y\_cross;  rec\_results.y\_train = \*y\_train;  rec\_results.Labels = LabelsVector;  return(rec\_results);  }  //  // END Generig Engine Class definitions  //  //  // BEGIN Generic Linear Engine (Virtual class)  //  inline float GenericLinearEngine::NRMSE(VectorXd y\_hat, VectorXd y)  {  float maxmin = y.maxCoeff()-y.minCoeff();  return(RMSE(y\_hat, y) / maxmin);  }  inline float GenericLinearEngine::RMSE(VectorXd y\_hat, VectorXd y)  {  long nr\_obs = y.size();  VectorXd errors = (y-y\_hat);  if (VERBOSE\_ENGINE)  {  debug\_info("Errors (last 3):");  debug\_info(errors.tail(3));  }  double sqNorm = errors.squaredNorm();  return(sqrt(sqNorm / nr\_obs));  }  inline void GenericLinearEngine::add\_cost(double J)  {  if (nr\_batches == 0)  {  // first use :)  J\_values = new VectorXd(1);  (\*J\_values)(nr\_batches) = J;  }  else  {  J\_values->conservativeResize(nr\_batches + 1);  (\*J\_values)(nr\_batches) = J;  }  nr\_batches++;  }  inline void GenericLinearEngine::init()  {  debug\_info("Generating object [" + CLF\_NAME + "]");  nr\_batches = 0;  Theta = NULL;  SingleClassTheta = NULL;  J\_values = NULL;  }  double myexp(double val)  {  return(exp(val));  }  MatrixXd& GenericLinearEngine::GetTheta()  {  return \*Theta;  }  string GenericLinearEngine::GetName()  {  return(CLF\_NAME);  }  double myround(double f)  {  return(round(f));  }  inline VectorXd GenericLinearEngine::PredictSingleClass(MatrixXd X)  {  VectorXd \*pred = new VectorXd(X.rows());  \*pred = X \* (\*SingleClassTheta);  return(\*pred);  }  inline MatrixXd GenericLinearEngine::Predict(MatrixXd X)  {  MatrixXd preds = X \* (\*Theta);  return(preds);  }  inline vector<string> GenericLinearEngine::PredictLabels(MatrixXd X)  {  MatrixXd y\_hat = Predict(X);  vector <string> PredictedLabels;  for (long i = 0;i < y\_hat.rows();i++)  {  int y\_hat\_idx;  y\_hat.row(i).maxCoeff(&y\_hat\_idx);  PredictedLabels.push\_back(LabelsVector[y\_hat\_idx]);  }  return(PredictedLabels);  }  inline vector<string> GenericLinearEngine::PredictLabelsUsingYHat(MatrixXd y\_hat)  {  vector <string> PredictedLabels;  for (long i = 0;i < y\_hat.rows();i++)  {  int y\_hat\_idx;  y\_hat.row(i).maxCoeff(&y\_hat\_idx);  PredictedLabels.push\_back(LabelsVector[y\_hat\_idx]);  }  return(PredictedLabels);  }  inline float GenericLinearEngine::TrainEvaluationSingleClass(bool bClass)  {  double dResult = 0.0f;  VectorXd y = \*y\_train;  MatrixXd X = \*X\_train;  long nr\_train = y.size();  if (SingleClassTheta == NULL && Theta == NULL)  return (dResult);  VectorXd y\_hat = PredictSingleClass(X);  long nr\_obs = y\_hat.size();  if (VERBOSE\_ENGINE)  {  debug\_info("Train Y\_Hat vs. Y\_train (last 3)");  MatrixXd result(nr\_train, 2);  result << y\_hat, y;  debug\_info(result.bottomRows(3));  }  if (bClass)  {  VectorXd y\_hat\_Rounded = y\_hat.unaryExpr(ptr\_fun(myround));  long positives = 0;  for (long i = 0;i < nr\_obs;i++)  {  if (y\_hat\_Rounded(i) == (y)(i))  positives++;  }  dResult = (double)positives / nr\_obs;  }  else  {  dResult = NRMSE(y\_hat, y);  }  return (dResult);  }  inline float GenericLinearEngine::CrossEvaluation(bool bClass)  {  double dResult = 0.0f;  VectorXd y = \*y\_cross;  MatrixXd X;  if (!bBiasAdded)  X = \*X\_cross;  else  X = X\_cross->rightCols(NR\_FEATS);  long nr\_cross = y.size();  if (Theta == NULL)  return (dResult);  MatrixXd y\_hat = Predict(X);  long nr\_obs = X.rows();  if (VERBOSE\_ENGINE)  {  MatrixXd result(nr\_cross, y\_hat.cols() + 1);  result << y\_hat, y;  debug\_info("Cross Y\_Hat vs. Y\_cross (last 5):",result.bottomRows(5));  }  if (bClass)  {  vector <string> preds = PredictLabelsUsingYHat(y\_hat);  long positives = 0;  for (long i = 0;i < nr\_obs;i++)  {  string predicted = preds[i];  string label = LabelsVector[(int)y(i)];  if (predicted == label)  positives++;  }  dResult = (double)positives / nr\_obs;  }  else  {  dResult = -1;  }  return (dResult);  }  inline float GenericLinearEngine::TrainEvaluation(bool bClass)  {  double dResult = 0.0f;  VectorXd y = \*y\_train;  MatrixXd X;  if (!bBiasAdded)  X = \*X\_train;  else  X = X\_train->rightCols(NR\_FEATS);  long nr\_cross = y.size();  if (Theta == NULL)  return (dResult);  MatrixXd y\_hat = Predict(X);  long nr\_obs = X.rows();  if (VERBOSE\_ENGINE)  {  MatrixXd result(nr\_cross, y\_hat.cols() + 1);  result << y\_hat, y;  debug\_info("Train Y\_Hat vs. Y\_train (last 5):", result.bottomRows(5));  }  if (bClass)  {  vector <string> preds = PredictLabelsUsingYHat(y\_hat);  long positives = 0;  for (long i = 0;i < nr\_obs;i++)  {  string predicted = preds[i];  string label = LabelsVector[(int)y(i)];  if (predicted == label)  positives++;  }  dResult = (double)positives / nr\_obs;  }  else  {  dResult = -1;  }  return (dResult);  }  inline float GenericLinearEngine::CrossEvaluationSingleClass(bool bClass)  {  double dResult = 0.0f;  VectorXd y = \*y\_cross;  MatrixXd X = \*X\_cross;  long nr\_cross = y.size();  if (SingleClassTheta == NULL && Theta == NULL)  return (dResult);    VectorXd y\_hat = PredictSingleClass(X);  long nr\_obs = y\_hat.size();  if (VERBOSE\_ENGINE)  {  debug\_info("Cross Y\_Hat vs. Y\_cross (last 3)");  MatrixXd result(nr\_cross, 2);  result << y\_hat, y;  debug\_info(result.bottomRows(3));  }  if (bClass)  {  VectorXd y\_hat\_Rounded = y\_hat.unaryExpr(ptr\_fun(myround));  long positives = 0;  for (long i = 0;i < nr\_obs;i++)  {  if (y\_hat\_Rounded(i) == y(i))  positives++;  }  dResult = (double) positives / nr\_obs;  }  else  {  dResult = NRMSE(y\_hat, y);  }  return (dResult);  }  //  // END Generic Linear Engine virtual class  //  //  // BEGIN Normal Regressor class definitions  //  void NormalRegressor::Train(MatrixXd X, MatrixXd y)  {  X\_train = new MatrixXd(X);  y\_train = new VectorXd(y);  Train();  }  template<typename \_Matrix\_Type\_>  \_Matrix\_Type\_ pseudoInverse(const \_Matrix\_Type\_ &a, double epsilon = std::numeric\_limits<double>::epsilon())  {  Eigen::JacobiSVD< \_Matrix\_Type\_ > svd(a, Eigen::ComputeThinU | Eigen::ComputeThinV);  double tolerance = epsilon \* std::max(a.cols(), a.rows()) \*svd.singularValues().array().abs()(0);  return svd.matrixV() \* (svd.singularValues().array().abs() > tolerance).select(svd.singularValues().array().inverse(), 0).matrix().asDiagonal() \* svd.matrixU().adjoint();  }  void NormalRegressor::Train()  {  debug\_info("Training: " + CLF\_NAME);  MatrixXd X = \*X\_train;  VectorXd y = \*y\_train;  MatrixXd xTx = X.transpose() \* X;  MatrixXd xT = X.transpose();  VectorXd TempTheta1(X.cols());  VectorXd TempTheta2(X.cols());  long duration1;  long duration2;  if (VERBOSE\_ENGINE)  {  // 1st solving with pseudo-inverse  high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();  MatrixXd xTxInv = pseudoInverse(xTx);  TempTheta1 = xTxInv \* xT \* y;  high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();  duration1 = duration\_cast<microseconds>(t2 - t1).count();  // now second method  high\_resolution\_clock::time\_point t3 = high\_resolution\_clock::now();  TempTheta2 = xTx.ldlt().solve(xT \* y);  high\_resolution\_clock::time\_point t4 = high\_resolution\_clock::now();  duration2 = duration\_cast<microseconds>(t4 - t3).count();  //SingleClassTheta = new VectorXd(TempTheta1);  SingleClassTheta = new VectorXd(TempTheta2);  }  else  {  // now second method  TempTheta2 = xTx.ldlt().solve(xT \* y);  SingleClassTheta = new VectorXd(TempTheta2);  }  if (VERBOSE\_ENGINE)  {  debug\_info("X data features size = " + to\_string(X\_loaded->cols()));  debug\_info("Theta PInv = " + to\_string(duration1) + " microsec");  debug\_info("Theta ldlt = " + to\_string(duration2) + " microsec");  debug\_info("T1(pinv) T2(ldlt):");  MatrixXd comp(TempTheta1.size(), 2);  comp << TempTheta1, TempTheta2;  debug\_info(comp);  if (\*SingleClassTheta == TempTheta2)  debug\_info("Using Theta2");  else  debug\_info("Using Theta1");  }  }  //  // END Normal Regressor class definitions  //  inline void OnlineClassifier::SimulateOnlineTrain()  {  if (Theta != NULL)  delete Theta;  Theta = new MatrixXd(NR\_FEATS + 1, NR\_CLASSES);  Theta->fill(0); // reset Theta  long TEST\_DEBUG = 1000;  BeginTimer();  for (long i = 0;i < X\_train->rows();i++)  {  MatrixXd obs = X\_train->row(i);  VectorXd yi(1);  yi(0) = (\*y\_train)(i);  if (VERBOSE\_ENGINE)// && (i == TEST\_DEBUG))  {  std::stringstream ss;  for (size\_t i = 0; i < yi.size(); ++i)  {  if (i != 0)  ss << ",";  ss << yi[i];  }  debug\_info("Training "+to\_string(i)+" th example with y = " + ss.str(),obs);  }  MatrixXd xi;  if (bBiasAdded)  xi = obs.rightCols(NR\_FEATS);  else  xi = obs;  OnlineTrain(xi, yi);  if (VERBOSE\_ENGINE)// && (i == TEST\_DEBUG))  {  //long time\_cost = EndTimer();  //debug\_info("Total time = " + to\_string(time\_cost) + " ms");  debug\_info("y\_OHM (1 row): ",LastYOHM.topRows(1));  debug\_info("y\_hat (1 row): ",LastYHat.topRows(1));  debug\_info("error (1 row): ",LastYERR.topRows(1));  debug\_info("Gradient (2 rows): ",LastGrad.topRows(2));  debug\_info("J array las val: ",J\_values->tail(1), true);  debug\_info("Theta (2 rows): ",Theta->topRows(2));  //debug\_info();  }  }  }  //  // BEGIN Online Classifier definitions  //  // yi is index in VectorLabels  void OnlineClassifier::OnlineTrain(MatrixXd xi, VectorXd yi)  {  long nr\_rows = xi.rows();  long nr\_cols = xi.cols();  VectorXd bias(nr\_rows);  bias.fill(1);  MatrixXd TempX(nr\_rows, nr\_cols + 1);  TempX << bias, xi;  long m = nr\_rows; // for convenience  MatrixXd yOHM(nr\_rows, NR\_CLASSES);  yOHM.fill(0);  for (long i = 0;i < nr\_rows;i++)  {  for (long j = 0;j < NR\_CLASSES;j++)  // now assume LabelsVector is correctly constructed  // and yi[i] is index in that vector  if (yi(i) == j)  yOHM(i, j) = 1;  }  // now we have the one hot matrix lets start working !  MatrixXd y\_hat = Predict(xi);  double J = (1.0 / m) \* cross\_entropy(yOHM, y\_hat); // MUST add regularization  add\_cost(J);  MatrixXd error = yOHM - y\_hat;  MatrixXd Grad = (-1.0 / m) \* TempX.transpose() \* error; // MUST add regularization    \*Theta = \*Theta - (LearningRate \* Grad);  LastGrad = Grad;  LastYOHM = yOHM;  LastYHat = y\_hat;  LastYERR = error;  LastXObs = xi;  }  inline double OnlineClassifier::CostFunction()  {  return 0.0;  }  inline MatrixXd OnlineClassifier::Predict(MatrixXd X)  {  long nr\_rows = X.rows();  long nr\_cols = X.cols();    VectorXd bias(nr\_rows);  bias.fill(1);  MatrixXd TempX(nr\_rows, nr\_cols +1);  TempX << bias, X;  MatrixXd XTheta = TempX \* (\*Theta);  MatrixXd SM = softmax(XTheta);  return(SM);  }  inline MatrixXd OnlineClassifier::softmax(MatrixXd z)  {  MatrixXd SM(z.rows(), Theta->cols());  ArrayXXd arr(z);  // first shift values  arr = arr - z.maxCoeff();  arr = arr.exp();  //cout << z;  //cout << arr;  ArrayXd sums = arr.rowwise().sum();  arr.colwise() /= sums;  SM = arr.matrix();  return(SM);  }  double myclip(double val)  {  double eps = 1e-15;  if (val < eps)  return(eps);  else  if (val > (1 - eps))  return(1 - eps);  else  return(val);  }  inline double OnlineClassifier::cross\_entropy(MatrixXd yOHM, MatrixXd y\_hat)  {  //y\_hat = y\_hat.unaryExpr(ptr\_fun(myclip));  MatrixXd J\_matrix = (yOHM.array() \* y\_hat.array().log()).matrix();  double J = -(J\_matrix.sum());  return(J);  }  //  // END Online Classifier definitions  // |

## Tehnici bazate pe Retele Neurale Adanci

### Utilizarea CNN in segmentarea semantica

Retelele convolutionale adanci constituie principalul vector stiintific si tehnologic care avanseaza stadiul actual al tehnologiei si cercetarii in domeniul recunoasterii computerizate.

Reţelele convolutionale sunt modele de Machine Learning cu capabilitati de vizualizare puternice, care produce ierarhii de caracteristici. Intrinsec, retelele convolutionale adanci ne pot prezenta informatia vizualala pe diverse nivele de analiza semantica cu ajutorul nivelelor “ascunse” (hidden layers) de neuroni artificiali. Conform lucrarii stiintifice “**Fully Convolutional Networks for Semantic Segmentation**” publicata de Johnatan Long, Trevor Darell et al. se arata ca retelele convolutionale dense si complete se pot proiecta si construi astfel incat sa genereze predictii si inferente semantice complete la nivel de pixel – acesta fiind pasul natural al modificarii predictiei brute de la nivelul intregii imagini care este analiza pana la nivelul punctului grafic unitar (pixel). Astfel se poate asocia fiecare pixel individual cu o anumita forma bidimensionala sau tridimensionala.

### Utilizarea LSTM pentru sisteme expert de tip Bot

Retelele neurale de tip LSTM (Long Short Term Memory) simuleaza functionarea neuronilor biologici responsabili de stocarea informatiei pe termen scurt si lung cu ajutorul unei porti de scriere, stocare si rescriere.

In materialele studiate este demonstrat atat prin formalizare matematica la nivel de model abstract de Machine Learning cat si prin demonstratii experimentale ca retelele neurale adanci de tip LSTM pot fi antranate complet cu script-uri de tip call-center sau alte tipuri de conversatii realizate anterior intre actori umani. In urma antrenarii reteaua LSTM este capabila sa formuleze raspunsuri coerente si logice la intrebari/probleme descrise de un utilizator uman in limbaj natural – utilizand fraza/intrebarea in limbaj natural a operatorului uman ca si punct de plecare a analizei si predictiei raspunsului ideal.

Concret vom utiliza retele de tip LSTM in vederea realizarii unui Bot care sa permita unui utilizator uman sa descrie modul de functionare al aplicatiei informatice dorite iar sistemul Cloudifier.NET va putea formaliza in limbaje de programare notiunile intelese, analizate si proiectate de componentele LSTM are Bot-ului.

### Framework-ul state-of-the-art TensorFlow

Pentru implementarea tuturor algoritmilor de tip Deep Learning se va utiliza state-of-the-art actual in acest domeniu lansat de Google in anul 2016 – framework-ul TensorFlow impreuna cu framework-ul Keras care are scopul de a adauga facilitati de nivel inalt bibliotecilor TensorFlow. In continuare este prezentata sintaxa TensorFlow (descrisa in limbajul Python) urmata de sintaxa Keras.

|  |
| --- |
| import tensorflow as tf  import numpy as np  # Create 100 phony x, y data points in NumPy, y = x \* 0.1 + 0.3  x\_data = np.random.rand(100).astype(np.float32)  y\_data = x\_data \* 0.1 + 0.3  # Try to find values for W and b that compute y\_data = W \* x\_data + b  # (We know that W should be 0.1 and b 0.3, but TensorFlow will  # figure that out for us.)  W = tf.Variable(tf.random\_uniform([1], -1.0, 1.0))  b = tf.Variable(tf.zeros([1]))  y = W \* x\_data + b  # Minimize the mean squared errors.  loss = tf.reduce\_mean(tf.square(y - y\_data))  optimizer = tf.train.GradientDescentOptimizer(0.5)  train = optimizer.minimize(loss)  # Before starting, initialize the variables. We will 'run' this first.  init = tf.global\_variables\_initializer()  # Launch the graph.  sess = tf.Session()  sess.run(init)  # Fit the line.  for step in range(201):  sess.run(train)  if step % 20 == 0:  print(step, sess.run(W), sess.run(b))  # Learns best fit is W: [0.1], b: [0.3] |

### Framework-ul state-of-the-art Keras

|  |
| --- |
| **from** keras.models **import** Sequential  **from** keras.layers **import** Dense, Dropout, Activation  **from** keras.optimizers **import** SGD  model = Sequential()  *# Dense(64) is a fully-connected layer with 64 hidden units.*  *# in the first layer, you must specify the expected input data shape:*  *# here, 20-dimensional vectors.*  model.add(Dense(64, input\_dim=20, init='uniform'))  model.add(Activation('tanh'))  model.add(Dropout(0.5))  model.add(Dense(64, init='uniform'))  model.add(Activation('tanh'))  model.add(Dropout(0.5))  model.add(Dense(10, init='uniform'))  model.add(Activation('softmax'))  sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=**True**)  model.compile(loss='categorical\_crossentropy',  optimizer=sgd,  metrics=['accuracy'])  model.fit(X\_train, y\_train,  nb\_epoch=20,  batch\_size=16)  score = model.evaluate(X\_test, y\_test, batch\_size=16) |

# Anexa – Rapoarte lunare

## Raport stiintific lunar 1

**Raport stiintific**

**de cercetare-dezvoltare in cadrul Cloudifier SRL**

**Nr. 62/31.10.2016**

|  |  |
| --- | --- |
| **Nume proiect** | Platforma de migrare automatizată în cloud a aplicațiilor și sistemelor informatice clasice cloudifier.net |
| **Beneficiar** | CLOUDIFIER SRL |
| **Cod MySMIS** | 104349 |
| **Nr. iregistrare** | P\_38\_543 |
| **Director Proiect** | Andrei Ionut DAMIAN |
| **Activitate conform planului de proiect** | 1. Activităţi de cercetare-dezvoltare (cercetare industrială şi/sau dezvoltare experimentală) - 1.1 State-of-the-art |
| **Luna** | Octombrie 2016 |
| **Echipa de cercetare-dezvoltare** | Andrei Ionut DAMIAN  Octavian BULIE |
| **Descrierea activitatilor desfasurate activitatii** | In decursul acestei luni a inceput procesul de analiza a stadiului curent al tehnologiei in domeniul sistemelor de tip Machine Learning cu accent pe zona de Deep Learning, domeniu de cercetare in dezvoltare la nivel international ce a luat amploare deosebita in ultimii 5 ani.  Scopul analizei stadiului actual al tehnologiei este acela de a determina metodele cele mai moderne/actuale de realizare a predictiilor/inferentelor in imagistica – in particular in cazul proiectului CLOUDIFIER referindu-ne la analiza imaginilor captate in timp real in timpul functionarii aplicatiilor si implicit analiza automatizata cu ajutorul recunoasterii avansate de forme/imagini a aplicatiilor “legacy” in vederea transalatarii acestora automatizate.  In decursul lunii octombrie 2016 au fost analizate cele mai recente si avansate lucrari de cercetare fundamentala si industriala provenite de la cele mai prestigioase institute si universitati printre care enumeram:   * Caltech – California Institute for Technology * MIT – Massachusetts Institute for Technology * Stanford * University of Toronto * Harvard * University of Washington   Principalele zone analizate au fost:   * Metodele de tip Deep Learning bazate pe Retele Neuronale Convolutionale – Deep Convolutional Neural Networks * Metode de tip shallow learning pentru invatarea supervizata a structurilor si a elementelor de imagistica utilizand modele de invatare in timp real (online learning) * Cele mai moderne abordari in Deep Learning – Tensor Flow * Cele mai moderne abordari in shallow learning – Extreme Boosted Decision Trees / Random Forests - XGBoost   In decursul lunilor noiembrie si decembrie se va continua analiza conform graficului de implementare a proiectului cu accent pe urmatoarele:   * Determinarea unui algoritm ideal pentru identificarea primitivelor de interfata grafica (butoane, campuri, ferestre, texte statice, etc) si a pozitiei acestora in cadrul ecranelor interfetelor grafice * Determinarea unui algoritm de tip Machine Learning pentru generarea AUTOMATA de interfe grafice si cod sursa aferente pe baza schitelor facute manual pe suport de hartie, tabla, etc * Analiza TensorFlow * Analiza XGBoost * Analiza metode si propuneri pentru biblioteci interne * Analiza si testarea experimentala a mediilor de procesare numerica masiv paralela cu ajutorul GPU (tehnologiile bazate pe nuclee de calul masiv paralel CUDA) |

|  |  |  |
| --- | --- | --- |
| Perioada | Efort in ore-om | Descriere |
| 3.10.2016-  14.10.2016 | 160 | Selectia si analiza preliminara a celor mai importante lucrcari din domeniul recunoasterii de imagini cu ajutorul retelelor adanci neurale convolutionale (Deep Convolutional Neural Networks). A fost inceputa analiza state-of-the-art pe ultimile cercetari realizate de laboratoarele de cercetare ale Google in Inteligenta Artificiala – biblioteca TensorFlow |
| 17.10.2016-  21.10.2016 | 80 | Analiza XGBoost – actualmente cea mai puternica infrastructura si biblioteca de shallow learning bazata pe modele de tip ansamblu |
| 24.10.2016-  31.10.2016 | 96 | Inceperea efectuarii de teste experimentale pe modele arhitecturale simple bazate pe regresii logistice adaptate si optimizate online si retele neural cu conectare completa.  Testele s-au realizat dupa cum urmeaza:   * Python cu ajutorul:   + Sci-Kit-Learn   + Biblioteca dezvoltata intern in cadrul Cloudifier pentru regresii logistice avansate (OnlineClassifierEngine.py)   + Biblioteca de retele neurale cu conectivitate completa realizata in cadrul Cloudifier * C++ cu ajutorul bibliotecii de calcul numeric optimizat Eigen |

## Raport stiintific lunar 2

**Raport stiintific**

**de cercetare-dezvoltare in cadrul Cloudifier SRL**

**Nr. 98/29.12.2016**

|  |  |
| --- | --- |
| **Nume proiect** | Platforma de migrare automatizată în cloud a aplicațiilor și sistemelor informatice clasice cloudifier.net |
| **Beneficiar** | CLOUDIFIER SRL |
| **Cod MySMIS** | 104349 |
| **Nr. iregistrare** | P\_38\_543 |
| **Director Proiect** | Andrei Ionut DAMIAN |
| **Activitate conform planului de proiect** | 1. Activităţi de cercetare-dezvoltare (cercetare industrială şi/sau dezvoltare experimentală) - 1.1 State-of-the-art |
| **Luna** | Noiembrie 2016 |
| **Echipa de cercetare-dezvoltare** | Andrei Ionut DAMIAN  Octavian BULIE |
| **Descrierea activitatilor desfasurate activitatii** | In decursul acestei luni a fost continuat procesul de analiza a stadiului curent al tehnologiei in domeniul sistemelor de tip Machine Learning cu accent pe zona de Deep Learning, domeniu de cercetare in dezvoltare la nivel international ce a luat amploare deosebita in ultimii 5 ani. Scopul analizei stadiului actual al tehnologiei este acela de a determina metodele cele mai moderne/actuale de realizare a predictiilor/inferentelor in imagistica – in particular in cazul proiectului CLOUDIFIER referindu-ne la analiza imaginilor captate in timp real in timpul functionarii aplicatiilor si implicit analiza automatizata cu ajutorul recunoasterii avansate de forme/imagini a aplicatiilor “legacy” in vederea transalatarii acestora automatizate.  In decursul lunii noiembrie 2016 au fost analizate cele mai recente si avansate lucrari de cercetare fundamentala in vederea determinarii unui set de algoritmi de Machine Learning ideali pentru identificarea primitivelor de interfata grafica (butoane, campuri, ferestre, texte statice, etc) si a pozitiei acestora in cadrul ecranelor interfetelor grafice   * Determinarea unui algoritm de tip Machine Learning pentru generarea AUTOMATA de interfe grafice si cod sursa aferente pe baza schitelor facute manual pe suport de hartie, tabla, etc * Analiza TensorFlow * Analiza XGBoost * Analiza metode si propuneri pentru biblioteci interne * Analiza si testarea experimentala a mediilor de procesare numerica masiv paralela cu ajutorul GPU (tehnologiile bazate pe nuclee de calul masiv paralel CUDA)   In decursul lunii decembrie se va continua analiza inceputa in lunile octombrie si noiembrie conform graficului de implementare a proiectului. |

| **Perioada** | **Efort in ore-om** | **Descriere** |
| --- | --- | --- |
| 01.11.2016-  11.11.2016 | 144 | Analiza model arhitectural de Retea Neurala total conectata in Python pentru analiza elementelor de interfata grafica si testarea experimentala a acesteia pe seturi de data de imagistica. Analiza PyCUDA – infrastructura programabila pentru procesoarele de calcul numeric masiv paralel CUDA |
| 14.11.2016-  30.11.2016 | 208 | Testarea in regim CPU si GPU a celui mai recent framework de Deep Learning realizat de laboaratoarele de cercetare ale Google – TensorFlow |

## Raport stiintific lunar 3

**Raport stiintific lunar**

**de cercetare-dezvoltare in cadrul Cloudifier SRL**

**Nr. 112/9.12.2016**

|  |  |
| --- | --- |
| **Nume proiect** | Platforma de migrare automatizată în cloud a aplicațiilor și sistemelor informatice clasice cloudifier.net |
| **Beneficiar** | CLOUDIFIER SRL |
| **Cod MySMIS** | 104349 |
| **Nr. iregistrare** | P\_38\_543 |
| **Director Proiect** | Andrei Ionut DAMIAN |
| **Activitate conform planului de proiect** | 1. Activităţi de cercetare-dezvoltare (cercetare industrială şi/sau dezvoltare experimentală) - 1.1 State-of-the-art |
| **Luna** | decembrie 2016 |
| **Echipa de cercetare-dezvoltare** | Andrei Ionut DAMIAN  Octavian BULIE |
| **Descrierea activitatilor desfasurate activitatii** | In decursul acestei luni a fost continuat procesul de analiza a stadiului curent al tehnologiei in domeniul sistemelor de tip Machine Learning cu accent pe zona de Deep Learning si in particular a sistemelor de analiza si recunoastere bazata pe inteligenta artificiala a imaginilor.  In decursul acestei luni analiza stadiului curent al cercetarii fost fost axat in principal pe lucrarea stiintifica publicata recent de J. Long et al “Fully Convolutional Networks for Semantic Segmentation”, lucrare considerata actualmente state-of-the-art in ceea priveste metodele de recunoastere si segmentare a componentelor in cadrul imaginilor. Pentru referinta prezentam anexat un scurt rezumat in limba engleza a lucrarii de referinta.  Principalele puncte pe care le urmarim in cercetare sunt urmatoarele:   1. Determinarea metodelor optime bazate pe Deep Learning pentru recunoasterea si segmentarea (identificarea locatiei spatiale) a elementelor de interfata grafica pe care Cloudifier.NET va trebuie sa le translateze automatizat din aplicatiile legacy in aplicatiile din mediul cloud computing. 2. Aplicarea de metode simple bazate pe algoritmi de machine learning superficiali (regresie logistica, arbori de decizie, clasificare naiva bazata pe teorema lui Bayes, clusterizare cu analiza distantelor euclidiene) precum si metode de segmentare iterativa a imaginilor analizate cum ar fi metoda ferestrelor deplasate continuu (“ferestre alunecatoare” sau sliding-windows algorithm) |

| **Perioada** | **Efort in ore-om** | **Descriere** |
| --- | --- | --- |
| 01.12.2016-  09.12.2016 | 96 | Continuarea analizei metodelor de recunoastere a imaginilor prin CNN (Convolutional Deep Neural Networks) |
| 12.12.2016-  31.12.2016 | 224 | Analiza unui model arhitectural Alpha ce urmeaza a fi definitivat in cadrul activitatii 1.2 de cercetare. Sistemul/model arhitectural Alpha va consta in construirea unui model matematic predictiv care sa poate recunoaste elemente simple de interfata grafica de utilizator (meniu, buton, etc) si sa poata reda locatia si 1-2 alte atribute de baza ale acestora |

**Fully Convolutional Networks for Semantic Segmentation**

Evan Shelhamer, Jonathan Long, Trevor Darrell

(Submitted on 20 May 2016)

Convolutional networks are powerful visual models that yield hierarchies of features. We show that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, improve on the previous best result in semantic segmentation. Our key insight is to build "fully convolutional" networks that take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. We define and detail the space of fully convolutional networks, explain their application to spatially dense prediction tasks, and draw connections to prior models. We adapt contemporary classification networks (AlexNet, the VGG net, and GoogLeNet) into fully convolutional networks and transfer their learned representations by fine-tuning to the segmentation task. We then define a skip architecture that combines semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed segmentations. Our fully convolutional network achieves improved segmentation of PASCAL VOC (30% relative improvement to 67.2% mean IU on 2012), NYUDv2, SIFT Flow, and PASCAL-Context, while inference takes one tenth of a second for a typical image.

Comments: to appear in PAMI (accepted May, 2016); journal edition of arXiv:1411.4038

Subjects: Computer Vision and Pattern Recognition (cs.CV)

Cite as: arXiv:1605.06211 [cs.CV]