Nr. 172 / 31.01.2017

Cloudifier.NET: Migrarea automată a software-ului, traducerea și dezvoltarea pe baza modelelor predictive masive, paralelizate, îmbătrânite

Nr. Ver	Data	Autori	Companie	Proiect
1	08.01.2017	AIDamian, O. Bulie	Cloudifier SRL	Platforma de automatizare a migrației în cloud a aplicațiilor și sistemelor informatice clasice Cloudifier.NET Analiza modelelor euristice
2	31.01.2017	AIDamian, O. Bulie	Cloudifier SRL	Platforma de automatizare a migrației în cloud a aplicațiilor și sistemelor informatice clasice Cloudifier.NET Analiza modelelor euristice - învățarea mașinilor

Conținut

Abstract

Domeniul dezvoltării software-ului automatizat și generativ este unul din domeniile care au beneficiat de o atenție generoasă atât din partea oamenilor de știință din domeniul științific, cât și din cel academic, în ultimii 40 de ani. Odată cu progresele recente în domeniul modelelor de predare automată a mașinilor și al calculării paralele masive, este acum disponibil un nou set de resurse pentru comunitatea științelor informatice pentru a cerceta și a dezvolta noi metode de dezvoltare automată de software inteligentă și inovatoare. În sau pe hârtie, indicăm faptul că sarcina de dezvoltare a interfeței utilizator cu software-ul complet poate fi complet automatizată. Propunem un model de învățare a mașinilor prin conducte, augmentat cu procesare paralelă cu GPU, pentru dezvoltarea interfeței cu vizualizările, modelele și controlorii aferenți. În cele din urmă, demonstrăm că întregul proces de dezvoltare a aplicațiilor online bogate poate fi complet automatizat și livrat prin intermediul serviciilor web. Abordarea noastră propusă va fi utilizată într-un experiment științific pentru a genera experiențe de utilizare pe deplin funcționale, inclusiv modele și controlori, pe lângă vederile reale.

Introducere

Problemele de ultimă oră și de cercetare, cum ar fi clasificarea imaginilor [1] [2] și segmentarea semantică a imaginilor [3], oferă noi oportunități pentru diferite activități legate de învățarea mașinilor, variind de la recunoașterea unei pisici într-o imagine, În scena și definirea tuturor atributelor scenei, cum ar fi toate obiectele cu dimensiuni și poziții. Concursurile de recunoaștere a imaginii de prestigiu, cum ar fi ImageNet [4], demonstrează o rată uimitoare de îmbunătățire a ultimilor 10 ani, atât în ceea ce privește utilizarea rețelelor de învățare în profunzime, cât și a rețelelor neuronale deosebit de profunde, precum și utilizarea formării masive paralele de calcul a mașinii Modele de învățare - de la câștigătorul competiției din 2010 care a obținut o precizie de 72% până la câștigătorul competiției din 2015, cu o acuratețe de peste 96%.

Cu toate acestea, avansarea infrastructurilor hardware bazate pe GPU capabile să furnizeze masă paralelă de calcul pentru sarcinile de învățare de mașină la costuri foarte mici a fost exploatată doar de domeniul învățării profunde și a rețelelor neuronale profunde în special.

În lucrarea noastră vom propune mai multe abordări: (i) un model bazat pe servicii bazate pe servicii de tip web pentru implementarea modelului de pregătire și predicție pe backend-ul masiv de calcul paralel al GPU și (ii) o abordare a proiectării,

dezvoltării și implementării software-ului automat pentru modulele de experiență ale utilizatorilor Bazat pe un model de conducere superficială a conductei de mașini. Vom argumenta că modelul nostru de conducte propus va fi capabil să analizeze schițele de interfață naivă trasată de om la imagini raster sau vectoriale și până la ecrane capturate pentru re-inginerie sau sarcini de traducere software. Practic, modelul de ansamblu va fi capabil să îndeplinească mai multe sarcini, cum ar fi (a) traducerea software-ului automatizat al modulelor de experiență utilizatorilor de la aplicații moștenite în medii cloud, (b) dezvoltarea și implementarea automată a modulelor UX bazate pe schițe de designer digital, Dezvoltarea prototipurilor UX bazate pe intrarea directă de utilizator (desene de mână brute)

Munca și abordarea conexe

Activitatea noastră se referă la cele mai recente progrese atât în domeniul calculului de înaltă performanță, cât și al învățării în mașină. Întregul câmp al inteligenței artificiale tinde să devină tot mai mult despre îmbunătățirea predicțiilor și a modelelor predictive pe care Goldfarb et al argumentează în lucrarea "Gestionarea mașinilor". Modelarea predictivă poate fi acum utilizată pentru sarcini extrem de variate, de la construirea unui sistem expert de chat-bots [6], bazat pe rețelele neuronale recurente de lungă durată pe termen scurt care ar folosi capacitatea de scriere a limbajului natural până la recunoașterea melanomului în imagini dermoscopice [7].

După cum sa menționat anterior, progresele recente din 2010-2011 în domeniul procesării informatice, a capacității de stocare și de comunicare - și în special în domeniul prelucrării paralele masive numerice / științifice pe bază de GPU [8] - au adus o îmbunătățire extraordinară abordărilor existente sau în curs de desfășurare, Modele. Tehnologiile GPU, cum ar fi arhitectura NVIDIA Pascal, oferă la niveluri acceptabile de prețuri pentru o putere de calcul de înaltă performanță capabilă să livreze peste 10 TFLOPS de calcul paralel pe mai mult de 3500 miezuri computationale numerice. Utilizarea computerelor bazate pe GPU este în mare parte aplicată în procesul de învățare profundă - atât experimentarea cercetărilor, cât și dezvoltarea producției - oricât de puțin se acordă atenție folosirii resurselor de calcul paralel GPU în tehnicile de învățare superficială a mașinilor, cum ar fi arborii decizionali, Și regresie logistică, modele Naive-Bayes. Cu toate acestea, progresele actuale și scorurile de predicție de ultimă generație obținute prin experimente științifice profunde de învățare dau puțin loc pentru dezvoltarea științifică și avansarea tehnicilor cunoscute de modele superficiale. Chiar mai mult, companii precum Microsoft, Google sau IBM propun în prezent motoare

cadru online, care permit crearea și experimentarea pe bază de modele de predicție bazate pe web [9] care sunt utilizate în întregime pentru experimentele de formare off-line și predicție, Infrastructuri.

Luând în considerare aspectele prezentate anterior, propunem un set de noi abordări moderne care ar putea împuternici potențialul modelelor de învățare a mașinilor superficiale cu capacitatea de a folosi resurse de calcul paralel bazate pe GPU. De asemenea, un alt aspect de sprijin pe care îl vom folosi în abordarea noastră se bazează pe cele mai importante caracteristici ale modelelor de învățare mecanică superficială: abilitatea modelului de a se auto-explica (deși în detrimentul puterii de predicție demonstrat de modele profunde cu masiv Ascunse și spații de greutate mai puțin explicative).

În restul lucrării vom aborda subiectele menționate anterior din trei perspective: (A) o abordare simplă pentru asigurarea eficientă a resurselor de calcul paralel bazate pe GPU pe baza serviciilor REST și (B) potențialele inovații de augmentare cu ajutorul calculului masiv paralel Tehnicile de învățare a mașinilor exterioare superficiale, în special în domeniul aplicațiilor de învățare on-line și al aplicațiilor de predicție în timp real cu (C) o abordare inovatoare a programării automate în timp real a modulelor UX bazate pe mașini superficiale superficiale, REST bazate pe abordarea de furnizare de calcul și modelele superficiale de învățare a mașinilor augmentate.

REST bazate pe servicii masive de calcul paralel

În prezent există mai multe opțiuni de programare disponibile pentru programarea motoarelor de calcul paralel GPU atât pentru scopuri științifice, cât și pentru inginerie / producție și probabil cele mai răspândite sunt OpenCL [10] și CUDA [8]. Practic, ambele modele menționate utilizează aceeași tehnică, cu doar mici diferențe. Cele două aspect fundamental în ceea ce privește abordarea de calcul constă în cele două entități centrale constând în *kernel - ul* - care este funcția care conține codul actual , care urmează să fie executate în paralel - și *dispozitivele* - dispozitivele de calcul reale care vor rula *kernel - ul* în paralel . În mod tradițional *kernel - ul* este o funcție descrisă de următorul algoritm generic:

Algoritmul 1 *Kernel (Data, BlockInformation)*

cid ∪ get coordonatele tuplu de la *BlockInformation*

Modificare *date* [Cid] cu greu codificate <*operație*>

Întoarcere

De notat că , în exemplul de mai sus *BlockInformation* definește segmentul de date reale pe care un anumit *dispozitiv* este de procesare în cadrul *datelor* și <*operație*> denotă operația codificate (compilat) că *kernel - ul* este de calcul pentru blocul particular al *datelor* primite. Pentru cazul particular al produsului dot matrice putem avea fiecare compute *dispozitiv* cu *kernel* produsul scalar a doi vectori (vectorul rând de 1 matrice st și vectorul coloană de 2 matrice nd).

$$A * B = A^T B = \sum a_i b_i \quad (1)$$

În cele din urmă avem următoarele două fragmente de cod pentru *nucleele* de calcul (1) , în OpenCL și CUDA pe bază de :

Cod 1 OpenCL Kernel pentru Matrix-Matrix produs scalar

```
__kernel void OpenCLMatrixDotKernel (const int A_ROWS,
                                     Const int BC_COLUMNS,
                                     Const int A_COLUMNS,
                                     Const __global float * A,
                                     Const __global float * B,
__ float global * C)
{{
Const int Row = get_global_id (0);
Const int Col = get_global_id (1);
Float acc = 0.0f;
Pentru (int c = 0; c <A_COLUMNS; c ++) {
Acc + = A [c * A_ROWS + rând] * B [Col * A_COLUMNS + c];
}
C [Col * A_ROWS + Row] = acc;
}
```

Cod 2 CUDA Kernel pentru Matrix-Matrix produs scalar

```
__global__ void CUDAMatrixDotKernel (Matricea A, Matricea
B, Matricea C)
{{
       Float acc = 0;
       Int rând = blocIdx.y * blockDim.y + threadIdx.y;
       Int col = blocIdx.x * blockDim.x + threadIdx.x;
       Dacă (rând> A.height || col> B.width)
              întoarcere;
       Pentru (int e = 0; e <A.width; ++ e)
              Acc + = elemente A. [rând * A.width + e] *
                     B. elemente [e * B. lățime + col];
       C. elemente [rând * C.width + col] = acc;
}
```

Așa cum am menționat anterior, în lucrarea de față vom prezenta o abordare REST [11] pentru furnizarea bazată pe cloud computing a serviciilor de calcul paralel. Această abordare va permite implementarea și consumul unui serviciu web care va avea următorul algoritm generic bazat pe alocarea FIFO

Algoritmul 2 *ParallelProcessingREST (Request)*

Pentru fiecare Kernel, perechea de date din Solicitare
Kernel αυα lua nucleu codul sursă de la *Cerere*
Datele de la ⇐ obține date CCD *Solicitare* de valoare sau prin referință la sursa externă
Disponibil ⇐ obține dispozitive disponibile de la cluster existent pe baza **algoritmului** adaptiv 3
Dacă este *disponibil* este valid apoi
Off-încărcare *Kernel* și *date* pe dispozitive *disponibile*
Ia - off-încărcare *Rezultat* și adăugați la *răspunsul*
Altfel
Răspuns ⇐ timpul Overhead depășește timpul de calcul
Răspuns de **retur**

Strategia actuală de execuție a unităților de procesare GPU a nucleelor descărcate se va baza pe un algoritm adaptiv care va lua în considerare sarcina serverului, complexitatea sarcinilor și sarcina / pregătirea GPU.

Algoritmul 3 ResourceAllocation *(Kernel, date)*

cFLOPS ⇐ evalua necesare FLOPS pentru *kernel - ului* și a *datelor*
Resursele ⇐ Compute necesare dispozitive bazate pe *cFLOPS*
ComputeTime ⇐ Aproximate timpul total de execuție pe baza *resurselor, cFLOPS*
WaitTime ⇐ Off-timpul de încărcare deasupra capului + timp până când dispozitivele sunt disponibile
Dacă *ComputeTime> WaitTime*
Rezultatul poziției ⇐ Coadă sau reale *dispozitive* alocate
Altceva
Alocarea *Rezultatul* ⇐ invalidată
Rezultatul de **retur**

Conductă de învățare a mașinilor superficiale extinsă

În abordarea noastră propusă pentru avansarea procesului de învățare a mașinilor superficiale prin amplificarea computerizată paralelă cu GPU, vom argumenta că conductele modelelor superficiale pot obține rezultate comparabile cu modele de învățare profundă pentru diverse aplicații. Deși majoritatea cercetărilor de ultimă oră în domeniul învățării în mașină se axează pe modele de

învățare profundă, modelele clasice de învățare a mașinilor cu o structură superficială ascunsă a straturilor, care sunt două caracteristici principale pe care le lipsesc modelele adânci: capacitatea de auto-explicare și robustețea algoritmică Pentru medii online și în timp real. Pentru a continua, vom analiza pe scurt cazul general al învățării supravegheate.

În învățarea supravegheată avem următoarele elemente principale în procesul de învățare:

$$X \in S^{MxN} \tag{2}$$

$$y \in L^{M} \tag{3}$$

$$h(X) = hypotesis_{model}(x_i|\theta \in \Theta) = \hat{y} \tag{4}$$

$$\Lambda(\theta) = L_{model}(h, X, y) \tag{5}$$

$$R(\theta) = Regularization_{model}\,(h(x|\theta)) \tag{6}$$

$$\Omega(\theta) = \Lambda(\theta) + R(\theta)\,, \qquad \theta \in \Theta \tag{7}$$

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}}\, \Omega(\theta) \tag{8}$$

În plus, vom folosi următoarele ecuații pentru exemplificarea cazurilor generalizate:

$$L_{cross\ entropy} = -\sum y_i \ln(h(x_i)) \tag{5'}$$

$$h_{linear\ regression}\left(x^{(j)}\right) = \sum_{i=1}^{n} x_i^{(j)} \theta_i = \theta^T x^{(j)} \tag{4'}$$

$$X_{transformed} = \frac{X - \min X}{\max X} \quad (9)$$

În cazul în care *X* în (2) este setul de date de formare cu probe *M* și caracteristici *N* (predictorul), *y* în ecuația (3) este etichetele dataset cu probe *M* corespunzătoare *X* setul de date, *h (X)* în ecuația 4) este modelul funcţia ipoteza definită ca pe baza unei valori predictor de intrare set de date și o greutate dată stabilită dintr - un spaţiu de greutate *Θ* calculează o predicție *în spaţiul etichetei, ΛΘ* în (5) este o funcție pierdere definită de modelul ca în ecuația (5') , de exemplu , a funcției pierdere Softmax încrucișată entropie, *este o funcţie* de *regularizare şi* în *final ΩΘ este funcţia obiectiv real. Ca urmare, scopul procesului* de *învăţare în orice model* de *învăţare supravegheat este minimizarea funcţiei obiectiv (sau maximizarea* dacă Dacă este de fapt o funcție de câștig în locul unei funcții de pierdere) așa cum este definită de ecuația (8). Deși în cazul modelelor profunde complexitatea spațiului de greutate întrece de departe complexitatea spațiilor de greutate din cadrul modelelor de mică adâncime, setul de greutăți , *care se dovedesc a fi minimizarea funcţiei obiectiv* să *conţină* cu *uşurinţă şi* proprietăți *auto* -explanatory - cum ar fi în caz de regresie liniară (4') în cazul în care fiecare greutate *direct de* amenzi importanța reală a caracteristicii de predictor *dat fiind faptul că datele* de *intrare sunt normalizate / standardizate ca în ecuaţia (9).* Chiar mai explicativă este cazul modelelor de instruire a copacilor de decizie, în care vederea reală a modelului instruit oferă informații complete și ușor de înțeles pentru orice persoană, indiferent de fundal.

Până acum, modelele superficiale, datorită naturii lor, au fost total ineficiente în îndeplinirea unor sarcini complexe, cum ar fi instruirea în clasificarea imaginilor și predicția în timp real pentru imagini în care dimensiunile predictorilor și spațiile de greutate se situează în milioane sau zeci de milioane de caracteristici. Acesta este cazul unui model predictiv care ar trebui să analizeze o imagine a unei interfețe de utilizator și să determine comenzile, etichetele, butoanele etc. împreună cu locațiile, dimensiunile și orice alte atribute inferioare. Pentru cazul particular al automate UX recunoașterea modelului predictiv trebuie să analizeze un flux de astfel de imagini livrate (ca un film / flux de exemplu) dintr - o interacțiune reală între un utilizator și sistemul -reading un flux de ecrane de înaltă definiție la peste *predictor caracteristici pe ecran şi cel puţin 10 cadre pe secundă* de *câteva ore şi genuri* Ting în ieșiri în timp real , așa cum este descris de ***ieşire** 1*.

Ieșirea 1 JSON / NoSQL structură de date pentru producția de conducte *(exemplu real în **Output** 2)*

```
Rezultat = {
       [<Nume element șir>]: {
                              "TYPE": <valoare>,
                              "LABEL": <valoare>,
                              "COORD": {<valori>}
                              "SIZE": {<values>}
                              "ACȚIUNE": {<valoare>}
                              [<Nume de subrețea șir>
{[<definition>]}
                                     .....
              ]
}
```

De asemenea, în ceea ce privește inferența reală a scenei de imagine, trebuie să analizăm din zone mici, cum ar fi matricele de 16x16x3 pixeli pentru controale simple cu un singur glif, până la matrice de 64x16x3 pixeli sau mai mult pentru butoanele de interfață, pentru controale mai complexe ale experienței utilizatorilor, cum ar fi grilele și tabelele , Până la ferestre sau zone de lucru de aceeași dimensiune ca și rezoluția reală a cadrului.

Propus de mică adâncime mașină de învățare Arhitectura noastră de conducte va fi compus din: (a) strat *RESTSPC* - un model de execuție pentru calcule științifice bazate pe GPU de calcul paralel , așa cum este descris de capitolul anterior în lucrarea noastră (b) strat *RESTOLAS* - un model de învățare mașină de mică adâncime bazat pe o regresie Softmax on - line modificat cu funcția ipoteză definită în ecuația (10) , care se va folosi stratul *RESTSPC* pentru toate calculele atât pentru gradientul on - line de optimizare bazată coborâre și predicțiile; (c) date *RESTWIND* pre-preprocesare și modelul post-procesare care se va ocupa de analiza de bază reală a informațiilor de intrare prime bazate pe *RESTSPC,* hrana pentru animale datele preprocesate la *RESTOLAS* și apoi se pregătească ieșirea conductei așa cum este descris în ***ieșire** 1* și ***ieșire** 2;*

Pentru stratul de pre-procesare *RESTWIND* a conductei noastre am decis să ia un algoritm simplu de fereastră glisantă și aplică o abordare modificată pentru setarea noastră de calcul paralel bazat pe un strat *RESTSPC.* Forma originală a algoritmului de alunecare al ferestrei prezentată de algoritmul 4 a fost modificată de algoritmul prezentat în Algoritmul 5. Forma prezentată a algoritmului 5 este inspirată efectiv de straturile de convoluție din cadrul CNN de ultimă generație [1].

Algoritm 4 *SlidingWindows (ImageFrame, Model)*

Lățimea *Lățimea* ⇐ a *ImageFrame*
Înălțimea Înălțimea ⇐ de *ImageFrame*
Lățimea *ModelWidth* ⇐ a imaginii care urmează să fie analizate *Modelul*
Înălțimea *ModelHeight* ⇐ a imaginii care urmează să fie analizate prin *model*
Pixeli *mărimea pasului* ⇐ sărită prin fereastra glisantă, propusă de *model*
Pentru fiecare *Y* în intervalul 1 la *înălțime ModelHeight* cu pas *mărimea pasului*
Pentru fiecare *X* în intervalul 1 până la *Width-ModelWidth* cu pas *mărimea pasului*
ModelImage ⇐ Extract sub - imagine *(X, Y, X + ModelWidth, Y + ModelHeight)*
Executați *Model.Train* pe *Model.Predict* pe *ModelImage*
Întoarcere

Algoritmul 5 *SlidingWindows (ImageFrame, model, RESTSPC)*

Lățimea *W* ⇐ de *ImageFrame*
Înălțimea *H* ⇐ a *ImageFrame*
Lățime *MW* ⇐ imaginii care urmează să fie analizate *Modelul*
Înălțimea *MH* ⇐ a imaginii care urmează să fie analizate prin *model*
Pixeli *SS* ⇐ sărită prin fereastra glisantă, propusă de *model*
Sarcini ⇐ *((W-MW) / SS) * ((H-MH) / SS)*
Alocarea ⇐ de alocare a resurselor Solicitare din partea *RESTSPC* bazată pe *Sarcini, model*
Cerere *RESTSPC* rula in paralel locuri de muncă numărul total de *sarcini* cu *model* și *ImageFrame*
Întoarcere

După cum sa menționat, procesarea principală cerută de stratul *RESTOLAS* al conductei noastre experimentale este compus în

principal din optimizarea cerută de funcția de cost *definită prin ecuația (5').* Folosirea abordării Softmax asigură că pentru fiecare element de scenă vom avea o inferență probabilistică - fiecare element de scenă va primi un scor probabilistic cu probabilitate maximă împreună cu câteva alte opțiuni potențiale (de exemplu, o "etichetă" dedusă cu o probabilitate de 60% Ar putea fi propus și ca un "buton aplatizat" cu o probabilitate de 35%). Pentru cazul special de învățare on-line, presupunem că fiecare caz de instruire este alimentat de modelul de învățare a mașinii de către stratul de pre-procesare, iar un pas de coborâre cu gradient simplu este calculat pe baza ecuației ...

Gradient de coborâre pentru regresia softmax ..

Programare automată UX bazată pe segmentarea semantică a imaginii

Principala inovație propusă a lucrării noastre în domeniul programării automate este introducerea unei conducte de învățare a mașinilor care va fi capabilă să realizeze analiza și reconstrucția interfeței utilizator în timp real în cazul pornirii sistemelor moștenite în mediile Cloud computing sau predicția UX pentru calculator Sau machete desenate manual.

Motivul principal al utilizării unui model de învățare mecanică superioară în locul unei rețele clasice neuronale profilactice pentru recunoașterea imaginii se bazează pe ghilimele datelor de antrenament pentru fiecare element de interfață individuală utilizator în momentul experimentării sistemului. Pentru a instrui o rețea profundă de convoluție așa cum este descrisă de [1] [3] [2], am avea nevoie de un model pre-instruit sau de un set de date masiv de antrenament. Datorită naturii scenelor țintă pe care trebuie să le deducem, nu vom putea folosi modele pre-instruite pe imagini / fotografii. Ca rezultat, se furnizează un set de antrenament cu dimensiuni variabile, așa cum este descris de datele și exemplele de mai jos.

Nr	Nr Fete	Feat 1	Feat 2	...	Feat 500	Feat 501	...	Feat 9900	Feat 9901	...	Clasă
...	...	...	...	...	...	...	...	...	...	...	...
21	500	214	121	...	410	nul	...	nul	nul	...	buton

...	...	...	...	...	...	...	...	...	...	...	...
151	9900	0	250	...	17	17	...	0	nul	...	eticheta
...	...	...	...	...	...	...	...	...	...	...	...

Tabelul 1 - Eșantionarea datelor de antrenament

Conducta noastră de învățare a mașinilor va utiliza mai multe procesări și se va baza în întregime pe un model REST și un motor de calcul paralel GPU. Cel mai simplu model propus va fi un motor de regresie Softmax augmentat optimizat pentru învățarea on-line și predicție în timp real care va fi prezentat în următorul algoritm

Algoritmul 6 GPUSoftmax

- *Scurtă prezentare a obiectivului și abordarea*
- *De ce o conductă și este paralelizată:*
  - *Învățare on-line în timp real pentru auto-adaptare a modelului*
  - *Predicții în timp real pentru serviciul REST*
- *Prezentarea motorului modelului de conducte bazat pe*
  - *Fereastra glisanta*
  - *Recunoașterea imaginii*
  - *UX Vizualizați inferența bazată pe recunoașterea imaginii*
  - *Inferența modelului UX bazată pe inferența vizualizării UX*
  - *UX Controler inferență bazate pe UX View / Model* de *inferență*

Migrarea automată a aplicațiilor

Traducerea automată a aplicațiilor moștenite a reprezentat întotdeauna un domeniu de cercetare important pentru comunitatea de informatică. Cu fiecare sosire a unor noi limbi de programare, cadre sau limbi oficiale de modelare, omul de știință de calculator a urmărit cercetarea și dezvoltarea diferiților traducători de cod care ar migra automat din codul sursă al aplicației vechi în noul limbaj de programare. Cele mai reușite proiecte de cercetare și de producție de-a lungul anilor s-au desfășurat în domeniul generării de

coduri automate de la limbi oficiale de modelare și în domeniul traducerilor automate de coduri. Cu toate acestea, procesul de migrare a codului sursă a fost întotdeauna însoțit de procese de re-engineering care ar fi trebuit să optimizeze mecanica internă a sistemelor software nou-migrate. Un alt obiectiv al traducerii software a fost și este totuși migrarea aplicației moștenite de la un mediu depreciat la unul mai modern, robust și mai fiabil - cum ar fi migrarea de la aplicații bazate pe desktop la client-server și apoi la mediile Cloud Computing. Cu toate acestea, în mai multe ocazii, datorită lipsei codului sursă veche, migrarea reală a fost forțată prin utilizarea terminalelor sau a desktop-urilor virtuale.

În lucrarea noastră propunem o arhitectură de model inovatoare concepută în special pentru provocările reprezentate în special de interfața cu utilizatorul și re-engineering-ul fluxului de date al aplicațiilor moștenite în care codul sursă nu este disponibil. Modelul nostru va folosi tehnici de învățare a mașinilor și în acest mod va putea "observa" și va deduce comportamentul funcțional al aplicației moștenite. La sfârșitul procesului inferențial, modelul nostru de conducte va fi capabil să genereze o definiție intermediară scenariată a aplicației moștenite observate. În cele din urmă, definițiile deduse și scripturi definite de propriul script de limbaj de definire intermediară (ITDL) vor fi utilizate de ultimul strat al modelului nostru de conducte pentru a genera codul țintă al platformei.

De la design mockup la software-ul functional UX

O prezentare a cazului de utilizare a aplicației moștenite din viața reală și ieșirea actuală ITDL a modelului va clarifica mai bine procesul actual descris în Algoritmul (1) - (5). În acest scop, vom folosi o aplicație desktop Windows reală prezentată mai jos în Figura (1). Din motive evidente de confidențialitate din Figura (1), detaliile / datele din unele câmpuri au fost obfuscate.

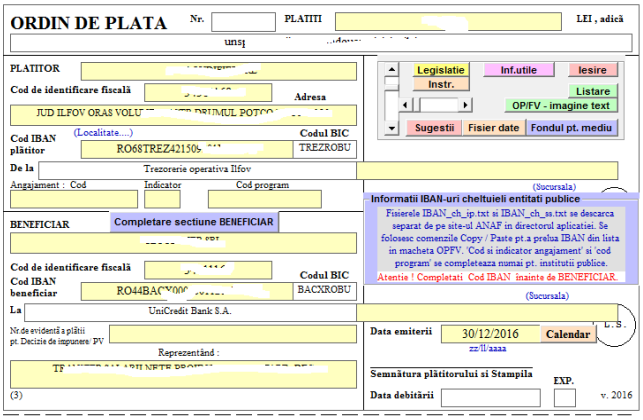

Figura 1 Aplicație moștenită de pe desktop

În următoarea ieșire (2) prezentăm una dintre inferențele de scenă generate de modelul nostru de conducte:

Ieșire 1 *Utilizați Cauza ITDL ieșire*

```
Rezultat = {

"LblORDIN": {

                "TIP": {

                                "CAPȚIUNE": 0,97

                                },

                "LABEL": "ORDIN DE PLATA",

                "COORD": {X: 5, Y: 5};

                "SIZE": {X: 150, Y: 20};

                "ACȚIUNE": "Niciuna",

                "BOUNDDATA": "Niciuna",

                "COPII": "Niciuna"

                                },

"LblNr": {

                "TIP": {

                        "AUTOEDIT": 0,55,

                        "EDIT": 0,41,

                },

                "LABEL": "Nr.",

                "COORD": {X: 165, Y: 4};

                "SIZE": {X: 40, Y: 20};
```

```
                "ACȚIUNE": "Niciuna",
                "BOUNDDATA": "lblNr_Data",
                "COPII": "Niciuna"
                        },
"LblPLATITI": {
                "TIP": {
                        "EDIT": 0,89
                        },
                "LABEL": "PLATITI:",
                "COORD": {X: 215, Y: 4};
                "SIZE": {X: 180, Y: 20},
                "ACTION": "on_lblPLATITI_Change"
                "BOUNDDATA": "Niciuna",
                "COPII": "Niciuna"
                        },
"LblLEIadica": {
                "TIP": {
                        "LABEL": 0,91
                        },
                "LABEL": "LEI, adică",
                "COORD": {X: 395, Y: 5};
                "SIZE": {X: 35, Y: 20};
                "ACȚIUNE": "Niciuna",
                "BOUNDDATA": "Niciuna",
                "COPII": "Niciuna"
                        },
"GidUnnameGridArea":
{{
                "TIP": {
                        "VIEWGRID",: 0,99
                        },
                "LABEL": "Niciuna",
```

```
                "COORD": {X: 5, Y: 35};

                "SIZE": {X: 450, Y: 350};

                "ACȚIUNE": "Niciuna",

                "BOUNDDATA": "Niciuna",

                "COPII": {

        "LblPLATITOR": {

                "TIP": {

                        "COMBOEDIT": 0,59,

                        "AUTOEDIT": 0,30,

                        "EDIT": 0,10

                        },

                "LABEL": "PLATITOR:",

                "COORD": {X: 3, Y: 3},

                "SIZE": {X: 185, Y: 20};

                "ACTION": "on_lblPLATITOR_Change"

                "BOUNDDATA": "lblPLATITOR_Data",

                                }

. . .

}
```

După cum este prezentat în *Output 1* modelul nostru de conducte generează o inferență completă scena pentru aplicația bazată pe desktop cu ecran capturate în Figura 2. Pe baza interacțiunii utilizatorului real cu sistemul de moștenire conducta generează o probabilitate maximă de probabilitate pentru fiecare clasă de utilizator de interfață de control. Clasa de control a interfeței utilizator implicită este prezentată împreună cu alți candidați potențial potențiali potențiali, dacă există.

Protocoalele UX bazate pe intrări naive

Nu vom continua cu o scurtă prezentare a unui potențial experiment cu interfață de utilizator naiv trase de mână așa cum este prezentat în Figura 2 - trase de mână mockup interfață .

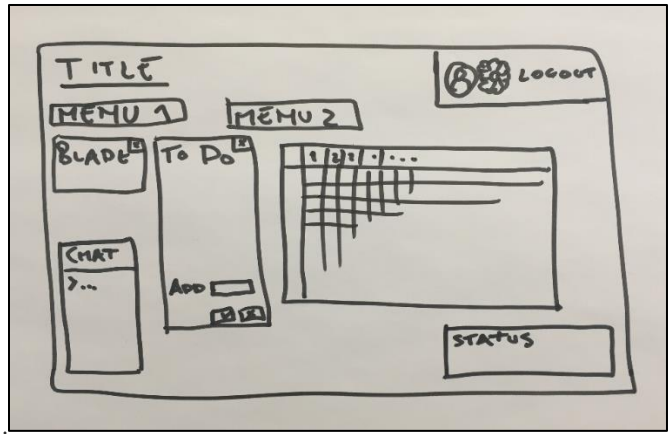

Figura 2 - macheta de interfață desenată manual

Modelul nostru de conducte pentru inginerie automată UX va fi capabil să aplice atât segmentarea semantică, împreună cu recunoașterea vizuală de bază a dat pentru a obține informații descriptive de ieșire care pot fi ușor traduse într-un limbaj de programare țintă. Mai jos este un fragment al producției efective a exemplului tras de mână de mai sus naiv:

Ieșire 2 HandDrawnInterface

```
Rezultat = {
              "LblTITLE": {
                           "TIP": "CAPTION",
                           "LABEL": "TITLE",
                           "COORD": {X: 10, Y: 10}
                           "SIZE": {X: 70, Y: 20}
                           "ACȚIUNE": "Niciuna"
                           }
              "MnuMenu1": {
                           "TYPE": "MENU",
                           "LABEL": "MENIU 1",
                           "COORD": {X: 10, Y: 50}
                           "SIZE": {X: 75, Y: 15}
                           "ACTION": "mnuMenu_Route1"
                           }
              "MnuMenu2": {
                           "TYPE": "MENU",
                           "LABEL": "MENIUL 2",
                           "COORD": {X: 110, Y: 50}
                           "SIZE": {X: 75, Y: 15}
                           "ACTION": "mnuMenu2_Route1"
                           }
              "FrmTodo": {
                           "TIP":  {"FRAME":  0,53,  "BLADE":
0,47}
                           "LABEL": "Pentru a face",
                           "COORD": {X: 40, Y: 50}
```

```
                    "SIZE": {X: 60, Y: 100}
                    "ACȚIUNE": "stdActionOkCancel"
                    "CONTROLS": {
                          "EdAdd": {
                                "TIP": "EDIT",
                                "LABEL": "ADD";
                                COORD: {X: 5, Y: 80}
                                "SIZE": {X: 50, Y:
15}
                                "ACȚIUNE":
"edValidate"
                                "MODEL" : {
                          .....
```

Referințe

[1] A. Krizhevsky, I. Sutskever și G. Hinton, "Clasificarea ImageNet Deep convoluțională Networks," *Advances in sistemele* de *prelucrare* a *informațiilor neuronale 1097-1105,* 2012.

[2] L. ea Szegedy "Mergând mai aprofundate cu convoluții," *IEEE Conferința pe Computer Vision si Pattern Recognition (CVPR),* 2015.

[3] J. lung, E. Shelhamer și T. Darrell, "Rețele complet convoluțional pentru Segmentarea semantic," *IEEE model* de *analiză și Machine Intelligence ISSN: 0162-8828,* nr. Mai 2016, 2016.

[4] ImageNet, "Large Scale ImageNet vizuală Concurență Recunoaștere (ILSVRC)," în *www.image-net.org/challenges/LSVRC/.*

[5] A. Agrawal, J. Gans și A. Goldfarb, "Gestionarea Roboților" *HBr,* 2016.

[6] Vinyals, "Un model Neural conversațională," în *ICML adânc* de *învățare Workshop 2015,* 2015.

[7] C. V, Q.-B. Nguyen și S. Pankanti, „Ansambluri de învățare profundă pentru Recunoașterea Melanomul în dermatoscopie Imagini," *Jurnalul de Cercetare și Dezvoltare,* 2016.

[8] J. Ghorpade, J. Parande și M. Kulkarni, "PROCESAREA GPGPU IN ARHITECTURA CUDA," *Advanced Computing: An International Journal (ACIJ),* vol. 3, 2012.

[9] M. Bihis și S. Roychowdhury, "Un flux generalizat pentru multi-clasă și sarcini de clasificare binare: O abordare Azure ML," în *Big Data (Big Data), 2015 IEEE International Conference on* 2015.

J. Stone, D. Gohara și G. Shi, "OpenCL: O programare paralelă standard pentru heterogeni Computing Systems" , *în Computing Science & Engineering*, vol. 12, nr. 3, [10] 2010.

RT Fielding, "Stiluri arhitecturale și proiectarea arhitecturilor software bazate pe rețea", [11] University of California, Irvine, 2000.

Proiect ID: P_38_543, Nr. Ctr. 98 / 09.09.2016, MySMIS: 104349, Apel - POC-A1-A1.2.1-C-2015,