

UNIVERSITATEA POLITEHNICA BUCUREȘTI  
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE  
DEPARTAMENTUL CALCULATOARE



## Lucrare de disertație

Învățare federată în scenarii de mobilitate umană la exterior

Dorian-Alexandru Verna

**Coordonator științific:**

Conf. dr. ing. Radu-Ioan Ciobanu

**BUCUREȘTI**

2024

## CUPRINS

1 INTRODUCERE.....	2
1.1 Context.....	2
1.2 Problema.....	3
1.3 Obiective.....	4
2 STATE-OF-THE-ART.....	5
2.1 Mobilitate umană la exterior.....	5
2.1.1 Modele de mobilitate umană la exterior.....	5
2.1.2 Algoritmi de predicție a mobilității umane la exterior.....	6
2.2 Învățare federată.....	8
2.2.1 Soluții bazate pe eficientizarea comunicării.....	8
2.2.2 Soluții care consideră eterogenitatea sistemelor.....	11
2.2.3 Soluții care consideră eterogenitatea datelor.....	12
2.2.4 Soluții care respectă accesul la date private.....	13
3 METODOLOGIA DE CERCETARE.....	14
3.1 Proof of concept.....	15
3.1.1 Design.....	15
3.1.2 Date de intrare.....	15
3.1.3 Tehnologiile folosite.....	16
3.1.4 Limitări ale metodologiei.....	16
4 MODELAREA PROIECTULUI.....	17
4.1 Arhitectură.....	17
4.2 Algoritmi și metode folosite.....	19
5 CONTRIBUȚII TEORETICE / PRACTICE.....	22
6 METRICI DE ANALIZĂ A PERFORMANȚEI ȘI REZULTATE.....	23
7 CONCLUZII.....	24
8 BIBLIOGRAFIE.....	25

# 1 INTRODUCERE

## 1.1 Context

Dinamica modului în care fiecare individ își desfășoară activitățile în fiecare zi a implicat și încă are la bază ideea de mobilitate, de nevoie de deplasare între diferite puncte în spațiu astfel încât să își poată realiza scopul. În ziua de azi, societatea se diferențiază cu mult față de secolul trecut, deoarece majoritatea situațiilor implică atât deplasarea urbană, fie ea realizată cu ajutorul mijloacelor de transport în comun, automobilelor, sau chiar pietonal, cât și deplasarea interurbană, cu alte cuvinte, între mai multe orașe. Ritmul vieții este în creștere, iar oamenii sunt în căutare de eficiență și siguranță atunci când ne referim la mobilitate.

Totodată, mobilitatea umană a devenit și mult mai ușor de monitorizat odată cu creșterea utilizării telefoanelor mobile, a dispozitivelor de tip wearable și, nu în ultimul rând, prin folosirea autovehiculelor autonome. Desigur, există numeroase alte surse de informație de pe urma cărora se poate înregistra deplasarea indivizilor sau populațiilor, acestea fiind prezentate în Cap 2. Cu toate acestea, agregarea datelor legate de mobilitatea umană se poate face mult mai ușor cu ajutorul dispozitivelor mobile, acestea putând fi interconectate pentru a forma rețele de comunicație. Astfel, se pot implementa soluții bazate pe datele colectate pentru a vizualiza tendința de deplasare a diferitelor mase de oameni.

Există o serie de aplicații pentru care prezicerea mobilității umane la exterior (în exteriorul clădirilor) poate reprezenta un avantaj enorm, aceste aplicații contribuind la realizarea unui *smart city*. Cu alte cuvinte, sunt colectate diferite informații de la dispozitive împrăștiate într-o multitudine de puncte din oraș, iar aceste informații sunt apoi procesate de aplicațiile aferente. Câteva utilizări ale datelor bazate pe mobilitate umană la exterior ar fi: controlul epidemiilor, protecția mediului, plasarea echipajelor speciale în oraș în scopuri preventive, furnizarea de informații legate de trafic, redirectarea traficului în scopul decongestionării etc.

În ciuda informațiilor prezentate anterior, agregarea acestor date devine o problemă, deoarece implică trimiterea unor informații private ale indivizilor către o instanță centrală (un server) care va realiza agregarea și prezicerea mobilității. Conform normelor europene [1], stocarea și utilizarea datelor personale, inclusiv a locației telefonului mobil, spre exemplu, este interzisă fără confirmarea individului, motiv pentru care soluția prezentată anterior nu este una validă, aceasta nefiind în conformitate cu regulile specificate prin intermediul GDPR [2].

Cu toate acestea, predicția mobilității umane a ajuns la un nivel destul de avansat încât să implice modele statistice bazate pe ML și care să ofere rezultate cu o acuratețe bună. Acest fapt, alături de numeroasele soluții și studii realizate în domeniul calculului distribuit și a ceea ce numim *edge computing*, au facilitat apariția unui concept numit *federated learning*, care presupune antrenarea unor modele pentru predicție pe dispozitivele remote, spre exemplu telefoanele mobile, și îmbunătățirea unui model central pe baza modelelor

realizate de dispozitivele remote. Astfel, datele private vor fi păstrate pe dispozitivele utilizatorilor, iar ceea ce va fi distribuit va fi modelul care realizează predicția.

Astfel, contextul actual implică utilizarea tehnologiilor bazate pe conceptul de învățare federată pentru a contribui la dezvoltarea aplicațiilor care au în vedere utilizarea datelor legate de mobilitatea umană. Realizarea acestor soluții reprezintă un subiect tratat de o serie de articole științifice de cercetare, subiect care este în continuă dezvoltare.

## 1.2 Problema

Există o serie de probleme care pot fi rezolvate cu ajutorul aplicării învățării federate în cadrul scenariilor de mobilitate umană, câteva dintre acestea au fost enumerate în cadrul Cap. 1.1. Cu toate acestea, problemele care se intenționează să fie rezolvate cu ajutorul *federated learning* nu se limitează doar la exemplele menționate anterior. În cele ce urmează, vom analiza problemele majore care sunt tratate de soluția prezentată.

Poate unul dintre cele mai puternice argumente pentru a sprijini dezvoltarea algoritmilor de mobilitate umană la exterior își are rădăcinile într-o problemă cu care societatea s-a confruntat în ultimii ani, aceasta fiind pandemia de COVID-19. Provocarea care a apărut în acea perioadă a ținut de izolarea virusului și prevenirea răspândirii acestuia. În acest context, prezicerea fluxului maselor de oameni a jucat un rol foarte important, fiind esențial pentru autorități să realizeze ce zone să fie izolate.

O altă problemă cu care se confruntă societatea de astăzi este reprezentată de migrație. Este cunoscut faptul că există un flux considerabil de oameni care se orientează spre mediile urbane și împrejurimile acestora. Astfel, distribuirea resurselor, locurilor de muncă, a oportunităților din punct de vedere social-economic și politic depind de locațiile destinație ale acestor fluxuri.

Totodată, este necesar ca orașele să permită soluții pentru decongestionarea traficului. În principiu, soluțiile pentru această problemă sunt nevoite să prezică zonele unde gradul de circulație va spori considerabil de-a lungul zilei, permițând astfel recomandarea de noi rute, sau chiar asignarea unor mijloace de transport public adiționale în scopul fluidizării traficului.

De asemenea, este necesar ca administrația orașelor să poată prezice locurile unde este probabil să fie nevoie de plasarea de autospeciale în mod preventiv, fie că vorbim de echipaje de poliție, de ambulanță sau pompieri. Riscul de evenimente nedorite crește odată cu fluxurile mari de indivizi, motiv pentru care plasarea acestor echipaje este strâns legată de problema mobilității urbane a oamenilor.

Nu în ultimul rând, subiectul învățării federate în contextul mobilității umane la exterior reprezintă în continuare o provocare pentru comunitatea științifică, găsirea unor soluții și tehnici noi fiind încurajată. Astfel, avem de a face și cu problema găsirii unor soluții eficiente și competitive cu tehnicile studiate anterior.

### 1.3 Obiective

Există o serie de aplicații posibile ale soluțiilor de predicție a mobilității umane la exterior, acestea fiind concentrate pe rezolvarea problemelor enunțate anterior, dar și pe dezvoltarea studiilor realizate cu privire la cele mai eficiente metode de implementare a învățării federate și de sporire a preciziei algoritmilor ML implicați în învățare.

Primul obiectiv important este acoperit de capitolul 2 al lucrării de față și implică identificarea soluțiilor existente cu privire la subiectul învățării federate în scenariul de mobilitate prezentat. Multe tehnici adoptate de-a lungul timpului au fost derivate din cele anterioare, motiv pentru care este necesar să înțelegem informațiile deja cunoscute despre subiect astfel încât să putem propune noi abordări.

Al doilea obiectiv este concentrat pe studierea unor noi metode de a realiza învățarea federată într-un mod cât mai eficient. Această parte va implica înțelegerea, studierea și implementarea unor propuneri de topologii și protocoale de comunicare între dispozitive mobile și unul sau mai multe servere centrale care conțin modele globale de predicție.

De asemenea, este important să identificăm un algoritm de predicție al mobilității care să se plieze pe datele pe care le vom colecta în cadrul pașilor incluși în îndeplinirea celui de-al doilea obiectiv, și să îl implementăm. Totodată, este necesar să fie realizată o comparație cu algoritmi de tip *state-of-the-art* și să propunem noi metode și să studiem rezultatele ce ar putea să le ofere.

Nu în ultimul rând, identificarea și implementarea unei aplicații pentru algoritmul de învățare federată constituie un obiectiv ce oferă un rezultat concret ce poate avea un efect direct pozitiv asupra indivizilor. În acest sens, putem dezvolta o schiță și o implementare de bază a unei aplicații care să rezolve problema definită anterior ce ține de *urban planning*.

## 2 STATE-OF-THE-ART

Aşa cum am menţionat anterior, au fost luate în calcul numeroase soluţii pentru rezolvarea problemelor menţionate în Cap. 1.2. Cantitatea de studii efectuate atât pe partea de învăţare federată, cât şi în aria referitoare la mobilităţi umane la exterior este considerabilă, motiv pentru care acest capitol tratează nivelul actual al cercetării efectuate separat, pe de o parte pentru a descrie diferitele abordări ale realizării predicţiei mobilităţii umane, iar de cealaltă parte, pentru a prezenta metodele existente bazate pe *federated learning*.

### 2.1 Mobilitate umană la exterior

Mobilitatea umană a fost şi este subiectul multor studii, în cele ce urmează vom prezenta principalele modele de mobilitate pentru mobilitatea umană urbană care au fost determinate: modele de mobilitate raportate la populaţie, raportate la individ şi modele unificate. De asemenea, vom menţiona şi principalii algoritmi de predicţie a mobilităţii umane la exterior care includ metode bazate pe lanţuri Markov, pe compresia datelor, pe date temporale şi pe *Machine Learning*.

#### 2.1.1 Modele de mobilitate umană la exterior

Primul model pe care îl putem include în analiza noastră şi care este considerat cel mai de bază din punct de vedere al mobilităţii umane, fiind încadrat în categoria modelelor de mobilitate raportate la populaţie este chiar modelul gravitaţional, sau *gravity model*. Acesta este inspirat din legea gravitaţiei enunţată de Sir Isaac Newton. Acest model presupune că fluxul de indivizi între două locaţii este direct proporţional cu dimensiunea celor două locaţii şi invers proporţional cu distanţa dintre cele două oraşe. Modelul gravitaţional presupune totuşi o serie de aproximări grosiere, poate cel mai puternic argument împotriva acestuia este reprezentat de faptul că modelul presupune aceeaşi valoare a fluxului indiferent de sensul parcurs.

Alte două modele de mobilitate raportate la populaţie, mai avansate de această dată, sunt *intervening opportunity model* şi *radiation model*. Cel de-al doilea este o optimizare a modelului gravitaţional, care doar date geografice şi nu necesită o estimare iniţială a parametrilor. De cealaltă parte, *intervening opportunity model* a fost propus încă din anul 1940 şi susţine că probabilitatea de migraţie este puternic impactată de oportunităţile care se pot regăsi la destinaţie. Formula care defineşte acest model poate fi observată în formula următoare [11]:

$$T_{i,j} = p_i \frac{e^{-\alpha S_{ij-1}} - e^{-\alpha S_{ij}}}{1 - e^{-\alpha S}} \quad (1)$$

Din punct de vedere al modelelor care se raportează la individ, putem aminti aici următoarele abordări: *Brownian motion*, *Levy flight*, *Continuous Time Random Walk (CTRW)*, *Social-Based model*. Primul model, la fel ca cel gravitaţional, îşi are originea în

domeniul fizicii, de data aceasta referindu-se la asemănarea dintre mobilitatea umană și mișcarea particulelor într-un lichid. De cealaltă parte, *Levy flight* implică premisa că multe deplasări mai scurte ale unui individ sunt urmate de o deplasare mai lungă, termenul pentru aceasta fiind *migrație*. Cel de-al treilea model are la bază transformata *Fourier-Laplace* pentru a calcula caracteristicile unui salt în spațiu pe o anumită perioadă de timp. Wang *et al.* [11] descrie trei tipuri de modele asociate conceptului *Social-Based model*, primul dintre acestea fiind un model spațio-temporal care este bazat pe ideea de *co-occurrence*, frecvența întâlnirilor unor indivizi determinând statusul social al acestora. Al doilea model prezentat în această categorie este bazat pe proprietatea numită *entropie*, și se concentrează pe predicția conexiunilor sociale existente. Cel de-al treilea model este bazat pe seturi de date *Foursquare* și implică predicția deplasării în diferite orașe.

Ultimul tip de model de mobilitate umană este cel unificat, poate cel mai important este propus de Yan *et al.* [12], care susține că gradul de atracție al unui individ față de o locație este determinat atât de memoria sa, cât și de populația locației respective.

### 2.1.2 Algoritmi de predicție a mobilității umane la exterior

Odată ce am determinat modurile în care se poate modela mobilitatea umană la exterior, este necesar să facem o introducere a principalilor tipuri de algoritmi folosiți pentru predicția mobilității. Așa cum am menționat anterior, aceștia sunt divizați în patru mari categorii. În Figura 1 este redată o privire de ansamblu asupra algoritmilor existenți pentru predicția mobilității [11].

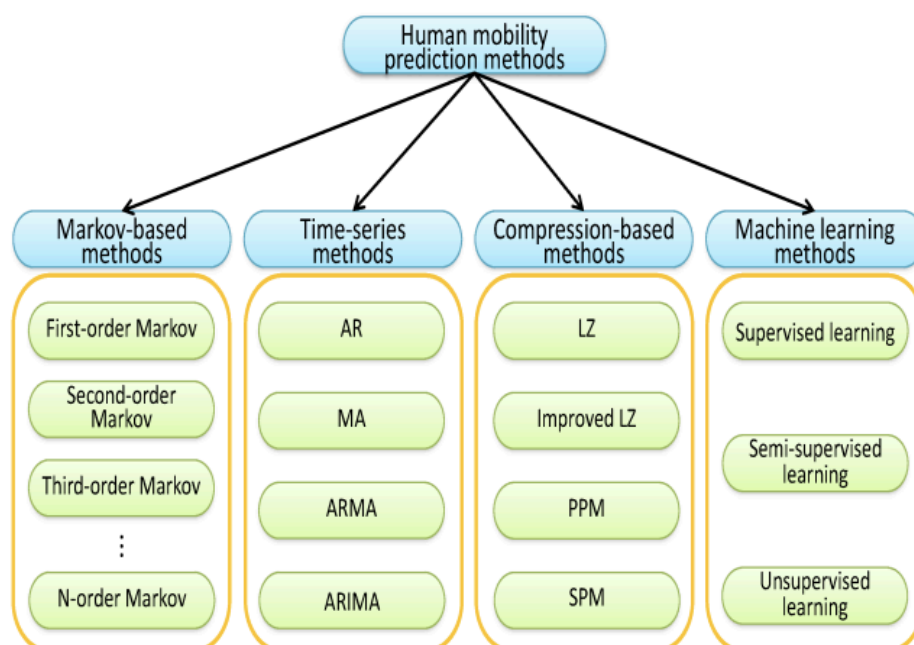


Figura 1. Taxonomia algoritmilor de predicție a mobilității umane [11]

Algoritmii de predicție a mobilității bazați pe lanțuri *Markov* reprezintă primul pas în determinarea mișcării umane în cadrul unui oraș. O particularitate a acestora este dată de

faptul că sunt specializați în a identifica probabilitățile de a face tranziția dintr-o anumită stare în alte stări. Algoritmii *Markov* de prim ordin iau în calcul doar starea prezentă, pe când, cei de ordinul 2 vor lua în calcul și starea imediat precedentă. Cu cât se mărește ordinul, cu atât vom considera mai mulți pași din trecut. Song *et al.* [13] realizează un studiu pe baza datelor colectate prin intermediul *Wi-Fi* de pe dispozitivele mobile prin care expune faptul că soluțiile bazate pe algoritmi *Markov* oferă rezultate în egală măsură și chiar mai bune comparativ cu alte metode mai complicate cum ar fi *LZ*, *PPM* sau *SPM*, atunci când ne referim la predicția următoarei locații de deplasare.

Cea de-a doua categorie de algoritmi sunt cei care au la bază compresia datelor, printre aceștia amintim *Lempel-Ziv (LZ)*, *Partial Matching (PPM)*, *Sample Pattern Matching (SPM)*. Poate unul dintre cele mai importante rezultate din punct de vedere al acestei categorii de algoritmi este prezentat de Alam *et al.* [13] prin intermediul algoritmului *SPEED (Sequence Prediction via Enhanced Episode Discovery)* care are o acuratețe de 88.3% și este implementat peste *PPM*.

Metodele de tipul *time-series* operează pe date care sunt de tipul *timestamped*, cu alte cuvinte, date care descriu evenimente ce sunt strâns legate de momentul de timp când s-au petrecut. Algoritmii din această categorie sunt concentrați pe folosirea unor diferite tehnici pentru a transforma aceste date în ceea ce numim *stationary data*, care implică lipsa variației datelor în funcție de timp. Metodele din această categorie includ *Autogressive (AR)*, *Moving Average (MA)*, *Autoregressive Moving Average (ARMA)*, *Autoregressive Integrated Moving Average (ARIMA)* [11]. Și în acest caz au existat studii ulterioare care au propus noi variații ale algoritmilor existenți, precum utilizarea seriilor de timp neliniare multivariate în contextul tiparelor de mobilitate deja existente pentru a spori acuratețea.

Nu în ultimul rând, de o importanță considerabilă sunt metodele bazate pe *Machine Learning*, acestea fiind folosite pentru a prezice atât volumul fluxurilor migrațiilor, cât și locația destinație, precum și momentul de timp din viitor când se consideră că prezicerile se vor concretiza. Și în acest caz putem identifica o serie de soluții propuse, acestea variind de la algoritmi ce presupun tehnici *deep learning* pentru date eterogene, la metode ce presupun folosirea unor concepte precum *Bayesian network* sau arbori de decizie. Un studiu important este realizat de Song *et al.* [14], care prezintă o abordare numită *DeepTransport*, algoritm *ML* care are la bază arhitectura *LSTM (Long-Short Term Memory)*. Figura 2 prezintă structura modelului *deep learning* folosit de *DeepTransport*.



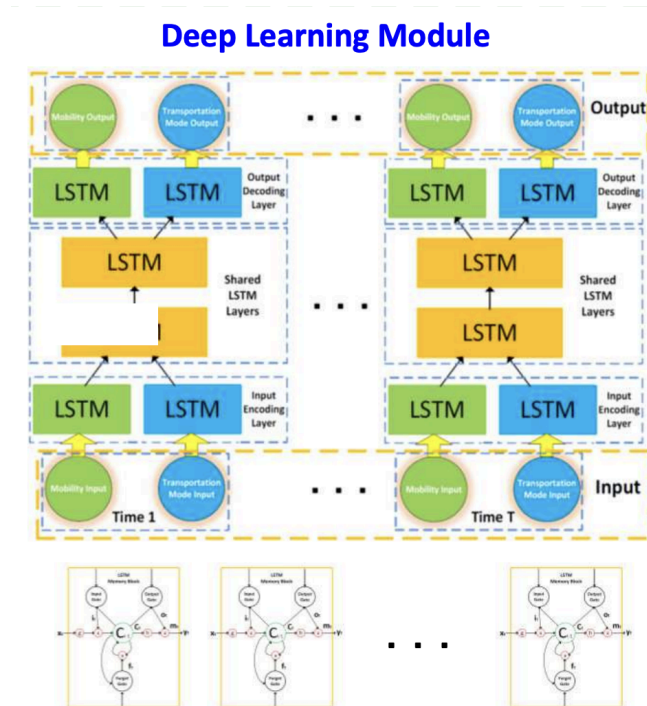


Figura 2. Arhitectura modelului *deep learning* în *DeepTransport*

## 2.2 Învățare federată

Conform prezentării anterioare a particularităților legate de învățarea federată, dezvoltarea soluțiilor pentru a asigura o predicție a mobilității trebuie să trateze provocările adresate anterior. Astfel, diverse abordări ale *federated learning* au fost studiate în funcție de diferite probleme adresate, în cele ce urmează fiind prezentate câteva dintre acestea într-un mod sistematic. În acest sens, conform studiului realizat de Li *et al.* [3], direcțiile pe care le vom aborda se împart în: soluții de învățare federată bazate pe eficientizarea comunicării, abordări care au în vedere eterogenitatea sistemelor angrenate în învățare, metode care iau în considerare eterogenitatea datelor și, nu în ultimul rând, metode care respectă politicile GDPR.

### 2.2.1 Soluții bazate pe eficientizarea comunicării

Primele soluții implementate aveau în vedere o tehnică numită *Minibatch Stochastic Gradient Descent* (SGD), deoarece metoda de optimizare a ponderilor din modele ML utilizate este bazată pe SGD sau variații ale acesteia. Cu toate acestea, în cazul Minibatch SGD, toate dispozitivele implicate în proces efectuează un pas de actualizare a gradientilor, după care rezultatele sunt imediat comunicate la serverul central pentru agregare și refacerea modelului de predicție global. Multiple epoci generează multiple reactualizări a gradientilor pe dispozitive, motiv pentru care se realizează o serie de pași în comunicare. Dezvoltări ulterioare au favorizat apariția unor algoritmi ca *FedAvg*, care este bazat pe un mecanism numit *Local Minibatch Stochastic Descent*, care implică transmiterea valorilor

gradienților de la fiecare dispozitiv după ce, în mod repetat, a fost realizată antrenarea modelului local, pentru un număr dat de pași, așa cum este descris și de Park *et al.* [4].

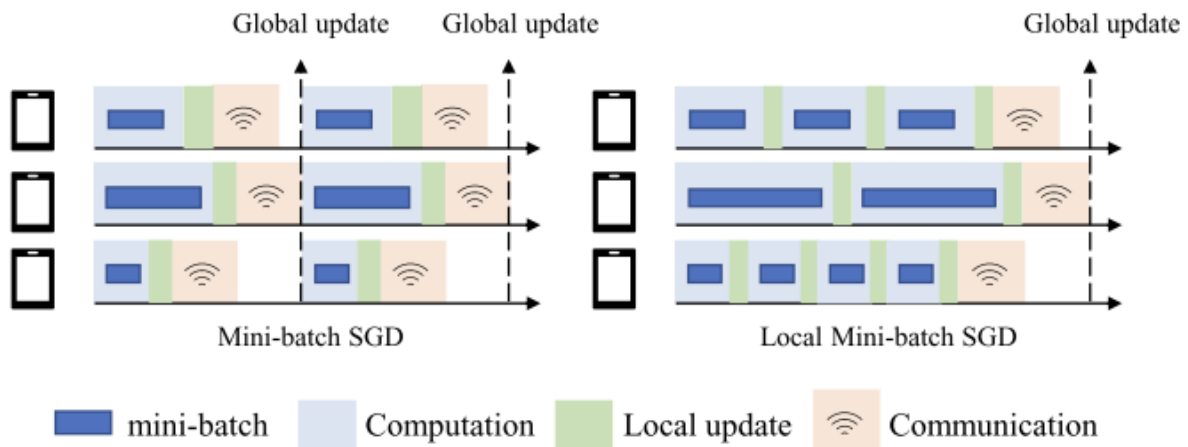


Figura 2. Minibatch SGD vs Local Minibatch SGD [4]

Același studiu prezintă o comparație între cele două metode, subliniind eficientizarea procesului de comunicare în cazul *FedAvg*, aspect ce poate fi observat și în Figura 2. Cu toate acestea, a doua soluție implică și un timp mai mare de procesare, dar și un consum mai mare de resurse pe dispozitivele locale, implicând astfel probabilitatea ca unele noduri să rămână în urmă din punct de vedere al procesului de *batch computing*, acestea fiind definite ca *stagers*, sau chiar să rezulte în eșecuri și chiar părăsirea rețelei pentru o perioadă de timp.

```

// All devices in K have same local minibatch size B
// and epoch size E
1:  $\beta \leftarrow (\text{split } P_k \text{ by size of } B)$ ;
2: for  $i$  in  $[1, E]$  do
3:   for  $b \in \beta$  do
4:      $w^k \leftarrow w^k - \eta_k \nabla F_k(w^k)$ ;

```

Figura 3. Procesul local de actualizare al *FedAvg* [4]

Pașii algoritmului *FedAvg*, sunt descriși într-o manieră simplistă în Figura 3. Așa cum se poate observa, prima oară este realizată o împărțire a setului de date localizat pe fiecare dispozitiv, după care se face un pas de actualizare a gradientilor pentru fiecare *batch*, timp de  $E$  epoci. După ce acest proces a fost realizat, dispozitivul comunică rezultatele serverului central.

Problema eficientizării comunicării este strâns legată și de dimensiunea datelor implicate în antrenarea modelelor și în transmiterea actualizărilor la serverul central. Există diverse studii efectuate cu privire la reducerea dimensiunii resurselor implicate în procesul de

învățare federată. Caldas *et al.* [5] prezintă două strategii pentru a eficientiza comunicarea server-client, acestea fiind *lossy compression* și *federated dropout*.

Conceptul de *lossy compression* se referă la compresia matricelor de ponderi calculate de fiecare dispozitiv local astfel încât să fie obținut un vector de valori căruia îi este aplicat un *basis transform*, trece prin două etape, *subsampling* și *probabilistic quantization*, și care este apoi trimis prin rețea spre a fi receptat de server. Serverul central aplică transformările inverse pentru a obține valorile vectorului și pentru a actualiza modelul cu rezultatele nou obținute.

*Federated dropout* presupune antrenarea unor submodele diferite pe fiecare dispozitiv și agregarea lor în cadrul serverului. Fiecare submodel are o arhitectura diferită, deoarece doar o parte din straturile de neuroni care se regasesc în modelul global se regăsesc și în submodel, fiecare dispozitiv având o distribuție diferită a straturilor.

Totodată, Konecny *et al.* [6] propune două metode de cuantificare a datelor pentru a reduce costul comunicării de tip *uplink*, acestea fiind bazate pe conceptele de *structured updates* și *sketched updates*. Primul concept implică parametrizarea actualizărilor diferitelor părți mai mici dintr-un model, pe când cel de-al doilea implică comprimarea întregului model.

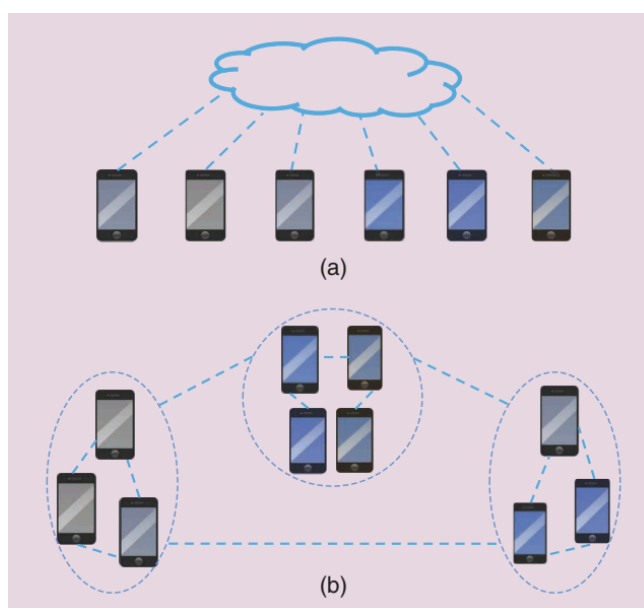


Figura 4. Topologie clasică vs. topologie descentralizată [3]

De multe ori comunicarea cu serverul central poate fi obstrucționată din diferite cauze, deoarece atât dispozitivele mobile, cât și serverele remote pot prezenta defecțiuni. Din această cauză au fost realizate diverse studii cu privire la schimbarea topologiei astfel încât să nu conțină un server central, ci mai multe clustere interconectate. He *et al.* [7] studiază abordări ce implică topologii descentralizate. Figura 4. [3] indică într-un mod simplist diferențele dintre cele două tipuri de topologii, ultima putând să prevină apariția unui *SPOF* (Single Point of Failure) prin intermediul serverului central și asigurând astfel reziliența

sistemului. Cu toate acestea, trebuie să avem în vedere faptul că o topologie descentralizată necesită și implementarea unor algoritmi de sincronizare la nivelul serverelor.

### 2.2.2 Soluții care consideră eterogenitatea sistemelor

Dispozitivele antrenate în procesul de învățare federată pot fi foarte diferite din punct de vedere al resurselor computaționale, motiv pentru care pot apărea probleme precum apariția unor noduri de tip *straggler*, care sunt caracterizate prin faptul că rămân în urmă în procesul de calcul și sincronizare a modelului de predicție cu cel al modelului global. Eterogenitatea acestor resurse computaționale este dată atât de *hardware*-ul diferit, cât și de conexiunea la internet care poate fi realizată prin intermediul unor tehnologii diferite (3G, 4G, 5G, Wi-Fi), care implică timpi de comunicare diferiți, și, nu în ultimul rând, de nivelul de încărcare al bateriei dispozitivelor.

O soluție este dată de implementarea comunicării asincrone cu serverul central. Actualizarea modelului global și trimiterea lui către o serie de dispozitive nu trebuie să fie realizată la momente specifice de timp. Lucrarea [8] studiază o metodă numită *Hogwild!* care implică realizarea actualizărilor pe partea serverului într-un mod *lock-free*, unde dispozitivele pot avea acces la aceleași părți din memoria serverului unde este stocat modelul global, putând să le modifice, în acest caz fiind introdus conceptul de *memory-shared systems*. Problema în cazul comunicării asincrone este dată de faptul că pot exista timpi de *delay* foarte mari atunci când vine vorba de resincronizarea modelelor.

Metode precum *active sampling*, sau selecția dispozitivelor client antrenate în comunicarea cu clientul reprezintă un punct important în problema eterogenității sistemelor. Nishio și Yonetani [9] realizează un studiu al algoritmului *FedCS (Federated Learning with Client Selection)*, unde protocolul algoritmului implică existența a doi pași, *Resource Request* și *Client Selection*. Primul pas presupune cererea de informații legate de capacitatea de calcul prezentă sau starea canalelor *wireless*, de la dispozitive client aleatoare. Al doilea pas folosește informațiile anterioare cu scopul de a determina ce dispozitive să comunice cu serverul pentru actualizarea modelului de predicție. O succesiune a pașilor prezentați se poate observa cu ușurință în Figura 5.

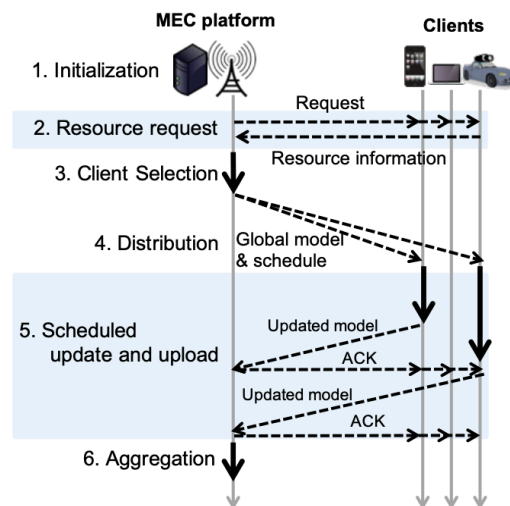


Figura 5. Privire de ansamblu asupra protocolului *FedCS* [9]

Toleranța la defecte este o direcție alternativă cu care poate fi tratată eterogenitatea sistemelor. În acest sens, a fost propusă ignorarea dispozitivelor care brusc părăsesc rețeaua și al căror proces de calcul rezultă în eșec, cum ar fi în cazul *FedAvg*. Desigur, acest fapt implică și faptul că modelul de predicție central va fi influențat de dispozitivele care au o funcționare bună și prezintă caracteristici *hardware* mai bune. Cu alte cuvinte, modelul va fi *biased* [3]. Algoritmul *FedProx* reprezintă o soluție la problema eterogenității, deoarece permite aplicarea unor actualizări în modelul central, care să fie conforme cu ceea ce poate dispozitivul să calculeze în funcție de capacitățile sale. Alte metode propuse implică replicarea datelor de antrenare pe alte dispozitive, dar aceste metode violează cerințele legate de protecția datelor personale.

### 2.2.3 Soluții care consideră eterogenitatea datelor

Datele colectate de dispozitivele mobile sunt diferite din multe puncte de vedere: în funcție de locație pot apărea diferențe majore din punct de vedere al limbii în care informațiile sunt stocate (deoarece mobilitatea umană nu implică doar coordonate, ci și denumiri de locații urbane, spre exemplu), sau din perspectiva cantității de date care sunt colectate, unele dispozitive înregistrând mai multe *data points* decât celelalte.

Și din acest punct de vedere au fost propuse mai multe soluții, acestea fiind bazate în principiu pe tehnici cum ar fi *metalearning* și *multitask learning*. Un exemplu de *framework* de optimizare pentru rezolvarea problemei eterogenității datelor este MOCHA [10], care presupune realizarea unor modele separate și specifice pentru fiecare dispozitiv și apoi utilizarea unei soluții bazate pe o reprezentare comună a acestora, pașii algoritmului fiind regăsiți în Figura 6.

```

1: Input: Data  $\mathbf{X}_t$  from  $t = 1, \dots, m$  tasks, stored on one of  $m$  nodes, and initial matrix  $\mathbf{\Omega}_0$ 
2: Starting point  $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$ ,  $\mathbf{v}^{(0)} := \mathbf{0} \in \mathbb{R}^b$ 
3: for iterations  $i = 0, 1, \dots$  do
4:   Set subproblem parameter  $\sigma'$  and number of federated iterations,  $H_i$ 
5:   for iterations  $h = 0, 1, \dots, H_i$  do
6:     for tasks  $t \in \{1, 2, \dots, m\}$  in parallel over  $m$  nodes do
7:       call local solver, returning  $\theta_t^h$ -approximate solution  $\Delta\boldsymbol{\alpha}_t$  of the local subproblem (4)
8:       update local variables  $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_t + \Delta\boldsymbol{\alpha}_t$ 
9:       return updates  $\Delta\mathbf{v}_t := \mathbf{X}_t\Delta\boldsymbol{\alpha}_t$ 
10:    reduce:  $\mathbf{v}_t \leftarrow \mathbf{v}_t + \Delta\mathbf{v}_t$ 
11:    Update  $\mathbf{\Omega}$  centrally based on  $\mathbf{w}(\boldsymbol{\alpha})$  for latest  $\boldsymbol{\alpha}$ 
12: Central node computes  $\mathbf{w} = \mathbf{w}(\boldsymbol{\alpha})$  based on the latest  $\boldsymbol{\alpha}$ 
13: return:  $\mathbf{W} := [\mathbf{w}_1, \dots, \mathbf{w}_m]$ 

```

Figura 6. Pașii algoritmului MOCHA

Desigur, există o multitudine de alte metode ca cea prezentată mai sus, cum ar fi modificarea topologiei astfel încât să fie sub forma unei *Bayesian network* sau folosirea unui algoritm prin intermediul căruia să se aleagă alternativ între folosirea modelului de predicție global, sau folosirea unui model antrenat local pe un anumit dispozitiv. Alte sugestii implică euristici care propun efectuarea unui anumit număr de actualizări în funcție de valorile obținute în urma aplicării funcțiilor *loss* pe algoritmi *ML* locali, sau folosirea unor tehnici de tipul *agnostic federated learning* care presupune folosirea unor scheme de optimizare bazate pe *mini-max*. De asemenea, se mai poate folosi și *q-FFL* atunci când ne referim la *objective function* pentru a calcula *loss*-ul și a determina gradul cu care fiecare model local va influența actualizările pe modelul global.

Din punct de vedere al conceptului de *non-i.i.d data* (*non independent and identically distributed data*), algoritmi ca *FedAvg* pot să divergă atunci când dispozitivele mobile realizează prea multe actualizări locale. Există și alte soluții, bazate pe abordări precum *parallel SGD* sau *FedProx*, ce au fost referite și anterior.

#### 2.2.4 Soluții care respectă accesul la date private

Conform cu cele prezentate anterior, respectarea normelor cu privire la datele cu caracter personal trebuie avută în vedere în contextul învățării federate. În acest sens, au fost propuse o serie de abordări, cea mai cunoscută fiind bazată pe *differential privacy approach*, aceasta implicând aplicarea unei perturbări în cadrul rezultatelor fiecărui pas intermediar de antrenare. Utilizarea *homomorphic encryption* reprezintă o altă metodă specifică *ML* atunci când vine vorba de conceptul de *privacy*. *SFE* (*Secure Function Evaluation*) și *SMC* (*Secure Multi-Party Computation*) reprezintă alte două metode folosite în acest caz, dar ele implică și existența unor costuri atunci când vine vorba de comunicare și de resurse computaționale necesare.

Conform studiului realizat de Li *et al.* [3], majoritatea abordărilor existente cu privire la învățarea federată folosesc variații ale soluțiilor bazate pe protocoale clasice criptografice cum ar fi *SMC* sau *differential privacy*. În Figura 7. putem observa cazul ideal pentru ceea ce numim *secure aggregation*. Serverul central reprezintă o entitate de încredere, care realizează criptarea datelor referitoare la algoritmi de predicție locali, cu ajutorul cărora se poate construi modelul global. Desigur, există posibilitatea ca utilizatorii să nu dorească partajarea informației de orice tip în format necriptat cu serverul, caz în care avem de a face cu un *untrusted server*. În acest caz, criptarea se face local pe fiecare dispozitiv, de data aceasta implicând un cost pentru trimiterea datelor criptate și agregarea lor pentru a obține un model. Cel de-al doilea caz poate fi vizualizat în Figura 8.

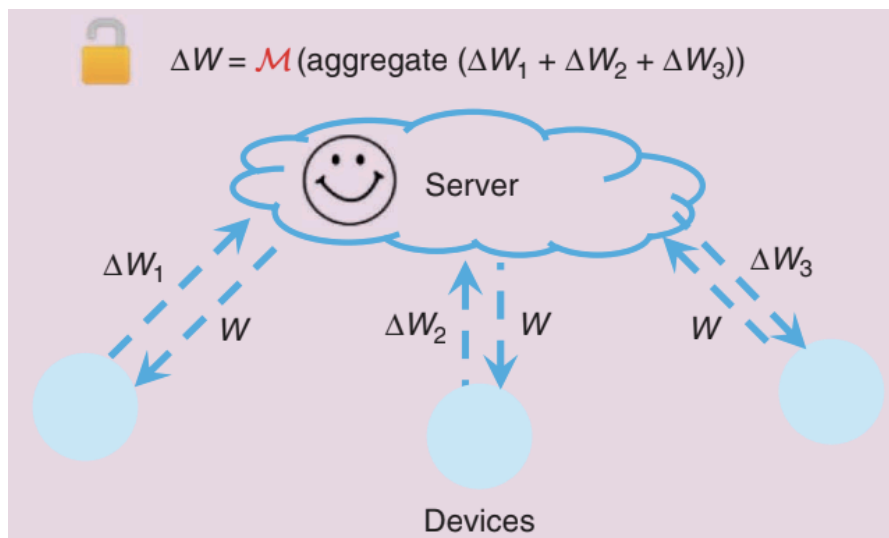


Figura 7. *privacy* la nivel global, *trusted server* [3]

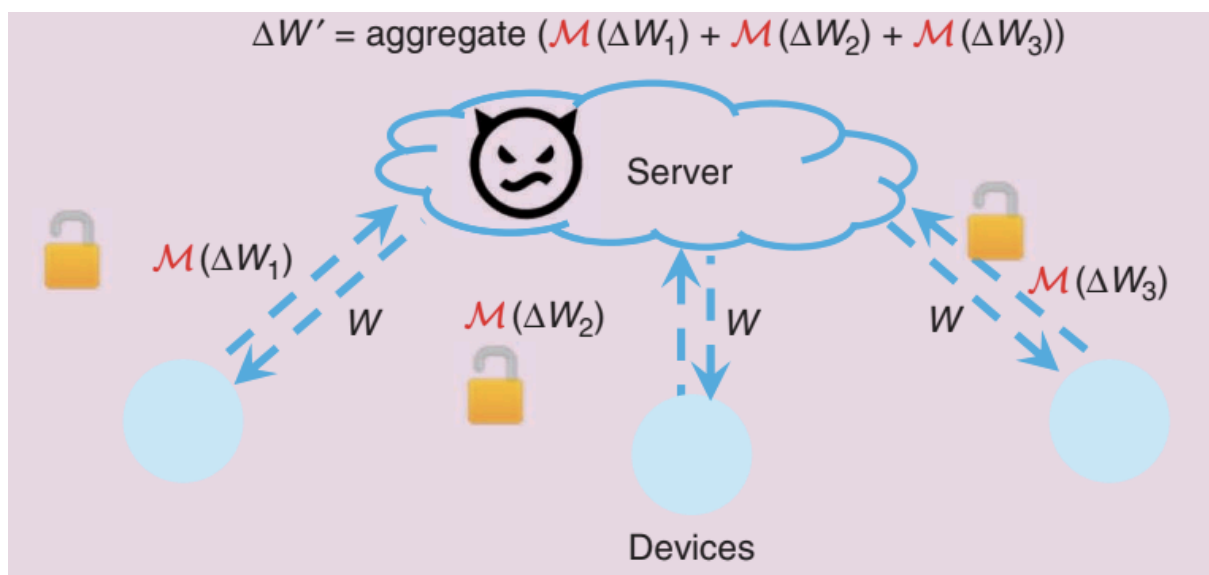


Figura 8. *privacy* la nivel local, serverul poate fi malițios [3]

### 3 METODOLOGIA DE CERCETARE

Din punct de vedere al metodologiei de cercetare ce a fost folosită, avem mai multe aspecte ce le putem lua în calcul, acestea fiind prezentate în mod gradual în cadrul acestui capitol, pornind de la prezentarea generală a motivației alegerii algoritmilor și metodelor folosite, până la întrebuițarea cunoștințelor și resurselor avute la dispoziție pentru a obține rezultatele dorite.

Prezentarea metodologiei se va realiza în funcție de etapele propuse în vederea obținerii unui rezultat final care să cuprindă un studiu amplu al subiectului prezicerii mobilității umane la exterior.

#### 3.1 Proof of concept

Primul milestone propus poartă numele de *proof of concept* și are în vedere realizarea unui prototip de topologie sumară, bazată pe un număr mic de entități care comunică între ele și care trimit datele acumulate local către un server care realizează un model de predicție perfecționat.

##### 3.1.1 Design

Partea de design este prezentată în detaliu în cadrul capitolului următor, în secțiunea 4.1. Entitățile menționate anterior sunt reprezentate de containere Docker, orchestrarea fiind realizată cu ajutorul unui fișier de configurație *docker compose*. Fiecare container comunică cu un server central și îi va trimite matrice de tranziție pentru a modela prezicerea mobilității cu ajutorul lanțurilor Markov.

Ca motivație pentru alegerea acestui prim pas în realizarea acestei lucrări, putem spune că am ales un algoritm de predicție relativ simplu față de metodele bazate pe Machine Learning, model care însă, în ciuda simplității, este recunoscut ca fiind eficient și ca fiind un etalon bun pentru obținerea unor rezultate care să reflecte adevărul. De asemenea, topologia folosită servește drept exemplu pentru posibilitatea realizării unei soluții care să implice o comunicare între mult mai multe dispozitive pentru îndeplinirea scopului prezentat.

Pentru prezicerea locației dispozitivelor avem în vedere folosirea unor date reale precum cele prezentate de Fu *et al.* [16], set de date publicat de Google, cu ajutorul căruia se dorește prezicerea cu precizie a poziționării dispozitivelor în diferite regiuni. Acest set de date va servi și în faza de validare, dar, pentru moment, în etapa de *proof of concept*, se dorește stabilirea unui set de date potrivit pentru testarea ulterioară a rezultatelor ce nu vor mai fi generate aleator.



### 3.1.2 Date de intrare

Datele de intrare, așa cum a fost menționat anterior, sunt generate în mod aleator, doar pentru a stabili dacă topologia este validă, și se poate realiza conexiunea pentru a testa agregarea matricelor și prezicerea poziției.

Totuși, am realizat și un studiu al setului de date prezentat în cadrul lucrării prezentate de Fu *et al.* [16] și am constatat că, odată realizată o clusterizare, se pot prezice regiunile în care se află diferite dispozitive la un moment dat, spre exemplu Mountain View, CA. Formatul datelor din acest set de date poate fi manipulat până ce putem ajunge la o reprezentare similară celei obținute în cadrul unui notebook realizat de Nanyu. T. S. [17]. Rezultatele pot fi observate și în Figura 9.

	phone	millisSinceGpsEpoch	latDeg	lngDeg
0	2020-05-15-US-MTV-1_Pixel4	1273608785432	37.904611	-86.481078
1	2020-05-15-US-MTV-1_Pixel4	1273608786432	37.904611	-86.481078
2	2020-05-15-US-MTV-1_Pixel4	1273608787432	37.904611	-86.481078
3	2020-05-15-US-MTV-1_Pixel4	1273608788432	37.904611	-86.481078
4	2020-05-15-US-MTV-1_Pixel4	1273608789432	37.904611	-86.481078

Figura 9. Coordonatele geografice ale unui dispozitiv în timp

### 3.1.3 Tehnologiile folosite

Cele două tehnologii folosite în cadrul acestui experiment sunt Docker și Flask, prima fiind folosită pentru crearea instanțelor de *worker* și pentru crearea nodului central, iar cea de-a doua, pentru implementarea propriu-zisă a transmițerii datelor, a creării matricelor de tranziție, și a agregării acestora.

Docker [18] reprezintă un produs software care se folosește de capacitatea de virtualizare a sistemului de operare pentru a livra aplicațiile în cadrul unor medii izolate care rulează pe sistemul de operare gazdă, acestea din urmă fiind numite containere [19].

Folosirea containerelor pentru a implementa serviciile descrise în cadrul arhitecturii sistemului indică o abstractizare a dispozitivelor *hardware* folosite de oameni, soluția prezentată fiind doar un experiment realizat local, pe un singur sistem de calcul.

Flask reprezintă un web framework bazat pe Python [20]. Flask poate fi folosit cu ușurință pentru a implementa și o arhitectură de tipul REST API. Totodată, Flask este un framework mai nou decât Django și este considerat a fi mai apropiat de standardele impuse de practicile comune Python (Pythonic [21]). O aplicație Flask nu conține foarte mult cod repetitiv sau care se poate evita [22] (*boilerplate*).

### 3.1.4 Limitări ale metodologiei

În ciuda argumentelor prezentate anterior cu privire la validitatea experimentului și la valoarea sa de *proof of concept*, trebuie să avem în vedere o serie de limitări pe care acesta le implică.

În primul rând, comunicația a fost realizată utilizând un număr mic de *containere* Docker, iar subiectul tratat în această lucrare implică comunicația între un număr mare de dispozitive mobile și serverul central. Creșterea numărului de noduri folosite va genera *overhead* în comunicație și e posibil ca o parte din mesaje să fie pierdute. Desigur, există metode care pot reduce costurile de comunicație, cum ar fi trimiterea selectivă a mesajelor de la diferiți *workeri* la nodul central, inclusiv folosirea mai multor noduri centrale astfel încât să folosim o topologie descentralizată.

Totodată, metoda care face uz de lanțuri Markov are rezultate bune atunci când ne referim la prezicerea mobilității umane la exterior. Multe lucrări științifice precum cea realizată de Qiao *et al.* [23] argumentează și propun algoritmi Markov pentru tratarea subiectului mobilității umane. Cu toate acestea, există algoritmi mult mai bine dezvoltați, care se bazează pe conceptul de *Machine Learning*, aceștia luând în considerare mai mulți parametri precum timpul când s-au realizat înregistrările, gradul de interes în locațiile definite de *basestation*-urile folosite, condițiile meteo, prezența unor evenimente sociale la anumite perioade de timp etc.

## 4 MODELAREA PROIECTULUI

Modelarea proiectului va fi realizată din două perspective, una fiind reprezentată de găsirea, replicarea și eventual îmbunătățirea unor algoritmi de predicție a mobilității umane la exterior, similari cu cei prezentați în capitolul anterior. În cadrul exemplului următor, care funcționează drept un prim *proof of concept*, ne-am concentrat pe realizarea unei soluții care să fie bazată pe lanțuri Markov și pe modelarea mobilității umane cu ajutorul unor matrice de tranziție care să reflecte deplasarea între diferite puncte din spațiu. Cea de-a doua perspectivă este strict legată de ideea de sisteme distribuite, unde, pentru început, voi avea în vedere realizarea unei simple configurații în Docker și mai apoi Kubernetes, care să implice un număr variabil de noduri care comunică cu un server central, ce are rol de agregare a informațiilor preluate de la fiecare nod și mai apoi transmiterea rezultatului agregat la nodurile de tip *worker*.

### 4.1 Arhitectură

Arhitectura proiectului descris în lucrarea de față constă într-o topologie de noduri care comunică cu un server central astfel încât fiecare nod contribuie la realizarea unui model de prezicere a mobilității umane la exterior. Această contribuție este realizată prin antrenarea unui model propriu fiecărui nod, transmiterea caracteristicilor algoritmului de prezicere

specific fiecărui nod către serverul central și realizarea unui *update* la nivel de nod cu caracteristicile modelului rezultat prin combinarea la nivel de server a modelelor anterioare.

Topologia experimentală folosită în primă fază este formată doar din 5 noduri, unul reprezentând serverul central, iar celelalte 4 reprezentând nodurile individuale care, în contextul lucrării, reprezintă dispozitivele mobile ale unor indivizi aleși în mod aleator, din cadrul unei populații. În Figura 10. putem observa o reprezentare elementară a topologiei folosite pentru a studia conceptul de învățare federată pentru a realiza un model care să poată prezice mobilitatea umană.

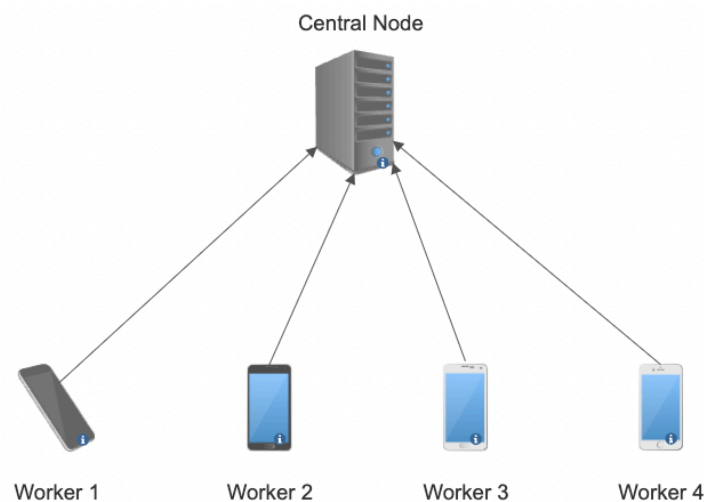


Figura 10. Topologia experimentală folosită drept *proof of concept*

Fiecare dintre cei 4 workeri este reprezentat de un container Docker, bazat pe o imagine construită pe baza unei aplicații Flask. De cealaltă parte, nodul central este și el reprezentat în același fel, tot printr-un server Flask.

Toate aceste instanțe create pe baza unor imagini Docker depind de o configurație realizată prin intermediul utilitarului *docker compose*. Mai jos se poate observa modul în care s-a realizat topologia de utilizată în cadrul experimentului inițial.

```
services:
  worker-node-1:
    image: worker-node:latest
    build: ./worker
    ports:
      - "5001:5000"
    depends_on:
      - central-node
    networks:
      - app-network
  worker-node-2:
    image: worker-node:latest
    ports:
      - "5002:5000"
    build: ./worker
    depends_on:
```

```

    - central-node
  networks:
    - app-network
worker-node-3:
  image: worker-node:latest
  build: ./worker
  ports:
    - "5003:5000"
  depends_on:
    - central-node
  networks:
    - app-network
worker-node-4:
  image: worker-node:latest
  build: ./worker
  ports:
    - "5004:5000"
  depends_on:
    - central-node
  networks:
    - app-network
central-node:
  image: central-node:latest
  build: ./central
  ports:
    - "5005:5000"
  networks:
    - app-network

networks:
  app-network:
    driver: bridge

```

Fiecare worker este definit separat, deoarece am dorit o separare a fiecărui *container* Docker, astfel încât să putem insista pe faptul că nodurile *worker* pot fi diferite atât din punct de vedere al caracteristicilor *hardware*, dar și din punct de vedere al *software*-ului disponibil pe acestea.

## 4.2 Algoritmi și metode folosite

Pentru dezvoltarea topologiei anterioare am utilizat o serie de cunoștințe generale legate de utilizarea sistemelor distribuite. De menționat sunt și metodele prin care au fost puse în aplicare ideile pe care s-a bazat experimentul de față, precum și algoritmi folosiți în cadrul acestuia.

Similar cu modul în care am prezentat arhitectura sistemului realizat în cadrul *proof of conceptului*, putem aduce aminti faptul că utilizarea unei topologii de tipul *Star* stă la baza configurației nodurilor prezente. Aceasta implică prezența serverului central și comunicarea fiecărui dispozitiv *worker* doar cu acest server. Această configurație asigură și o oarecare securitate în cadrul rețelei propuse, deoarece se garantează protejarea datelor cu caracter

privat ale fiecărui nod client, dat fiind faptul că serverul central este considerat a fi o entitate care nu poate avea un caracter malițios.

Tot din punct de vedere al algoritmilor și metodelor folosite în cazul lucrării de față, putem menționa și utilizarea REST API ca mod de comunicare între instanțele *containerelor* Docker utilizate. Astfel, un *worker* va trimite prin intermediul unui endpoint și a unui *request* de tipul POST, o matrice de tranziție care expune probabilitățile calculate pentru nodul respectiv de a se deplasa între mai multe locații din cadrul unui oraș. De cealaltă parte, nodul central va trimite și el la un anumit interval de timp câte un *request* de tipul POST către nodurile de tipul *worker*, urmând ca fiecare din aceste tipuri de noduri să facă o agregare la nivel local al noii matrice de tranziție obținută de serverul central. Mai jos se poate observa modul în care este implementată funcția ce permite trimiterea matricei de tranziție de la *worker* la nodul central.

```
def send_data_periodically(interval, size):
    """
    Send data to the central node at regular intervals.

    Args:
        interval (int): The interval in seconds at which data is sent.
        size (int): The size of the sample data list to generate.
    """
    while True:
        matrix = generate_markov_matrix(size)
        response = requests.post(f'{central_node_url}/aggregate', json={'matrices':
[matrix]})
        print(f'Sent matrix: {matrix}, Response: {response.json()}')
        time.sleep(interval)
```

Din punct de vedere al modalității alese pentru a prezice modalitatea umană, am propus o abordare inițială bazată pe lanțuri Markov, nivelul de dificultate în modelarea acestora fiind mai scăzut comparativ cu un model de *Machine Learning*. Algoritmii de predicție de dificultate superioară reprezintă un subiect important într-o parte ulterioară a lucrării.

Așadar, descrierea unui sistem ce trece prin diferite stări într-un interval de timp poate fi reprezentată cu ajutorul unui lanț Markov. Notabil este faptul că trecerea de la o anumită stare la o stare imediat următoare nu depinde de stările în care se afla sistemul inițial, ci doar de starea curentă. Tranzițiile între stările considerate sunt realizate cu ajutorul probabilităților, acestea fiind dispuse sub forma unei matrice care se numește *matrice de tranziție*. În formula (1) se poate observa modul în care este construită o astfel de matrice.

$$P = \begin{pmatrix} P_{AA} & P_{AB} \\ P_{BA} & P_{BB} \end{pmatrix} \quad (1)$$

Primul element al matricei reprezintă probabilitatea ca, sistemul aflându-se în prezent în starea  $A$ , acesta să își păstreze starea curentă. Al doilea element de pe prima coloană, spre exemplu, indică probabilitatea tranziției din starea  $A$  în starea  $B$ .

O proprietate cheie a acestor matrice de tranziție este reprezentată de faptul că suma probabilităților de pe fiecare rând este egală cu 1. O probabilitate de 0.2 de a trece din starea  $A$  în starea  $A$  indică o probabilitate de 0.8 de a trece din starea  $A$  în starea  $B$ , formula (2) fiind sugestivă pentru această proprietate.

$$\sum_j P_{ij} = 1 \quad (2)$$

Este foarte important să oferim și un exemplu prin care se poate prezice o stare ulterioară, având matricea de tranziție la dispoziție și starea curentă a sistemului. Să presupunem că ne aflăm în prezent în starea  $A$ . Acest fapt este descris prin intermediul vectorului  $[1 \ 0]$ . Primul element din cadrul vectorului corespunde stării  $A$ , iar cel de-al doilea corespunde stării  $B$ . Acestea fiind spuse, dacă la momentul  $t$  ne aflăm în starea  $A$ , atunci vectorul care indică probabilitățile de a ne afla în oricare dintre stările considerate la momentul  $t + 1$  va fi obținut prin realizarea produsului din formula (3).

$$v_{t+1} = v_t * P \quad (3)$$

De obicei, matricea de tranziție este considerată ca fiind constantă, pe baza acesteia fiind realizate predicțiile cu privire la stările ulterioare ale sistemului. În cazul experimentului nostru, vom genera aleator matricele de tranziție la fiecare pas, dar este important să menționăm cum ar fi acestea generate atunci când avem de a face cu dispozitive reale, care înregistrează în timp real poziția geografică prin intermediul coordonatelor.

În primul rând, matricile de tranziție pe care le vom folosi reflectă probabilitatea tranziție unui nod între două puncte geografice. Punctele geografice sunt reprezentate în mod normal prin coordonate geografice, dar aceste coordonate reprezintă o metrică prea exactă pentru dificultatea acestui *proof of concept* pentru a putea fi folosite. Acesta este motivul pentru care am ales o abstractizare a poziției care trebuie prezisă, folosind astfel zone în loc de coordonate, o zonă corespunzând unui grup de coordonate geografice grupate în funcție de distanță. Pentru simplitate, nu vom implica momentan coordonate geografice în soluția noastră, acest fapt urmând să fie implementat ulterior. De precizat însă, este faptul că, odată ce decidem folosirea coordonatelor, este important să realizăm o clusterizare a acestora pentru a le categorisi pe zone.

Acestea fiind spuse, vom introduce termenul *basestation (BS)* pentru a defini stările în care se află un sistem la un moment dat de timp. Spre exemplu, maparea unui oraș va fi realizată utilizând mai multe elemente de tip *BS*, care vor fi obținute după realizarea unei operații de clusterizarea, operație care poate fi realizată cu ajutorul unui algoritm de tipul *K Means* [15].

Un exemplu de matrice realizată de un nod *worker* este reprezentată în formula (4), unde 0.7 este probabilitatea tranziției în starea *BS1*, dacă starea curentă este tot *BS1*, 0.3 este probabilitatea tranziției în *BS2* din *BS1*, 0.4 definește tranziția din *BS2* în *BS1*, iar 0.6 specifică probabilitatea rămânerii în *BS2* dacă nodul era în *BS2* și înainte.

$$P = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix} \quad (4)$$

Există mai multe metode prin care se poate realiza agregarea matricelor, cele mai elementare moduri de a realiza acest aspect sunt următoarele:

- Realizarea unei medii aritmetice între elementele corespondente fiecărei matrice

Practic, această metodă presupune însumarea elementelor matricelor, și împărțirea sumei la numărul de matrice agregate. Această metodă este una validă și recomandată spre a fi pusă în practică atunci când nu poate fi determinată ponderea cu care contribuie diferite sisteme la colectarea datelor legate de locație, sau când nodurile contribuie în mod egal la antrenarea modelului global. Formula (5) pune în prim plan metoda prezentată în cadrul acestui paragraf.

$$P_{agg} = \frac{1}{n} \sum_{k=1}^n P_k \quad (5)$$

- Realizarea unei medii ponderate între elementele corespondente matricelor

Atunci când un dispozitiv are o înregistrează o mobilitate mai mare decât multe alte dispozitive, sau când datele colectate de acesta au o validitate mai bună decât în cazul altor dispozitive, este indicat să oferim o pondere mai mare măsurătorilor realizate de acest node, motiv pentru care realizarea unei medii ponderate ar fi recomandată atunci când ne referim la agregarea a două sau mai multe matrice de tranziție, aspect evidențiat și în formula (6).

$$P_{agg} = \frac{1}{n} \sum_{k=1}^n w_k * P_k \quad (6)$$

În acest caz,  $w_k$  reprezintă ponderea pentru matricea  $P_k$ , iar suma tuturor ponderilor este egală cu unitatea.

Configurația existentă folosește prima metodă, deoarece folosirea unor date randomizate oferă aceeași credibilitate pentru fiecare nod *worker* folosit.

## 5 CONTRIBUȚII TEORETICE / PRACTICE

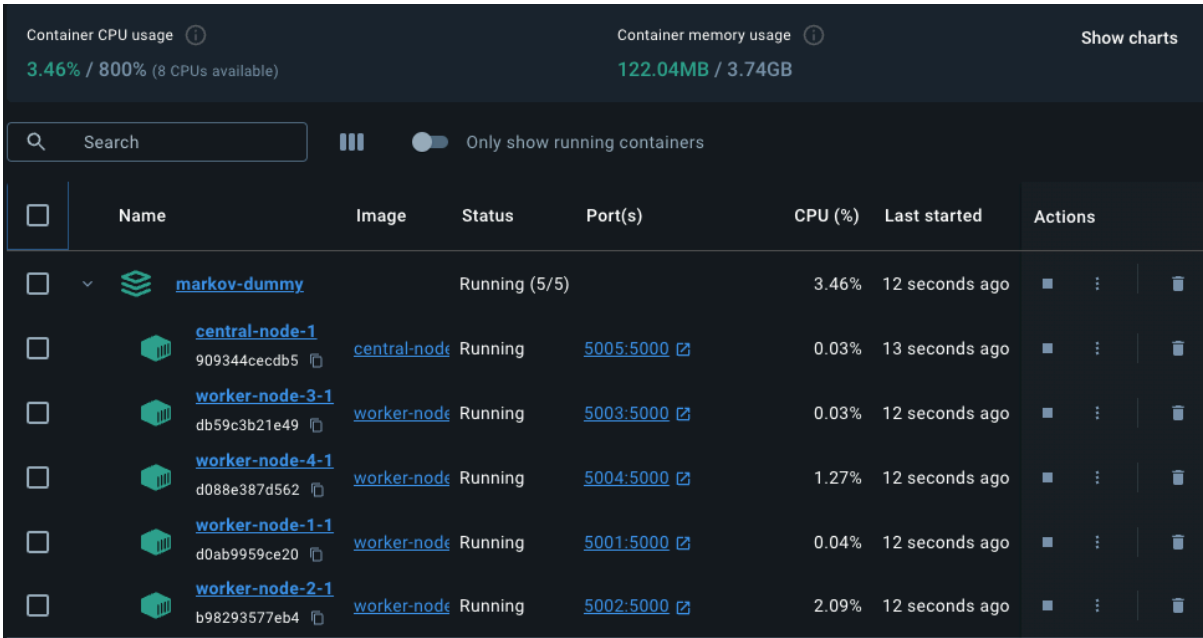
Din punct de vedere al contribuțiilor teoretice, am realizat o revizuire a literaturii existente legate de subiectul acestei lucrări, analizând o serie de lucrări științifice, mai ales în capitolul care corespunde analizei state-of-the-art. Totodată, descrierea unei topologii logice și a unei arhitecturi de tipul *proof of concept* au contribuit la realizarea porțiunii teoretice a lucrării.

Contribuțiile practice ale acestei lucrări includ dezvoltarea unui *proof of concept* pentru un sistem de prezicere a mobilității umane utilizând containere Docker și orchestrare cu Docker Compose. Prototipul demonstrează fezabilitatea și eficiența modelului propus, evidențiind capacitatea de scalare pentru a include multiple dispozitive. Alegerea unui algoritm de predicție simplu, dar eficient, oferă o soluție practică și accesibilă pentru industrie. Validarea modelului asigură o oarecare relevanță și aplicabilitate în scenarii reale, contribuind astfel la îmbunătățirea predicțiilor de mobilitate.

## 6 METRICI DE ANALIZĂ A PERFORMANȚEI ȘI REZULTATE

Analiza rezultatelor a implicat vizualizarea matricei rezultate, asigurarea performanței ansamblului de *containere* Docker prin verificarea constantă a *log*-urilor acestora, precum și revizuirea matricei finale rezultate în urma agregării spre a verifica validitatea acesteia.

Pentru a vizualiza crearea și buna funcționare a instanțelor *worker* și a nodului central, am utilizat interfața Docker Desktop, cu ajutorul căreia am putut vizualiza crearea topologiei, aceasta fiind expusă în Figura 11.



The screenshot shows the Docker Desktop interface. At the top, it displays 'Container CPU usage' at 3.46% / 800% (8 CPUs available) and 'Container memory usage' at 122.04MB / 3.74GB. Below this is a search bar and a toggle for 'Only show running containers'. The main table lists containers with columns: Name, Image, Status, Port(s), CPU (%), Last started, and Actions. The containers listed are 'markov-dummy' (Running (5/5)), 'central-node-1' (Running), 'worker-node-3-1' (Running), 'worker-node-4-1' (Running), 'worker-node-1-1' (Running), and 'worker-node-2-1' (Running).

























	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	 <a href="#">markov-dummy</a>		Running (5/5)		3.46%	12 seconds ago	  
<input type="checkbox"/>	 <a href="#">central-node-1</a> 909344ceadb5	<a href="#">central-node</a>	Running	<a href="#">5005:5000</a>	0.03%	13 seconds ago	  
<input type="checkbox"/>	 <a href="#">worker-node-3-1</a> db59c3b21e49	<a href="#">worker-node</a>	Running	<a href="#">5003:5000</a>	0.03%	12 seconds ago	  
<input type="checkbox"/>	 <a href="#">worker-node-4-1</a> d088e387d562	<a href="#">worker-node</a>	Running	<a href="#">5004:5000</a>	1.27%	12 seconds ago	  
<input type="checkbox"/>	 <a href="#">worker-node-1-1</a> d0ab9959ce20	<a href="#">worker-node</a>	Running	<a href="#">5001:5000</a>	0.04%	12 seconds ago	  
<input type="checkbox"/>	 <a href="#">worker-node-2-1</a> b98293577eb4	<a href="#">worker-node</a>	Running	<a href="#">5002:5000</a>	2.09%	12 seconds ago	  

Figura 11. Interfața unde pot fi regăsite instanțele de noduri folosite

Din punct de vedere al performanței, *containerele* ocupă aproximativ 122 de MB din memoria RAM totală aflată pe sistemul local de calcul. Astfel, putem afirma că sistemul



propus nu este foarte costisitor din punct de vedere al resurselor consumate, dar trebuie luată în calcul nevoia de resurse mai performante dacă se dorește implementarea la scară largă a unei soluții perfecționate.

Log-urile din cadrul fiecărui *container* pot fi vizualizate cu ușurință tot din această interfață, nodurile de tip *worker* vor printa matricile realizate și trimise mai apoi la nodul central, dar și nodul central va printa la rândul său mesajele rezultate de pe urma *request*-urilor de tipul POST făcute de nodurile *worker*. În Figura 12, se pot observa log-urile aflate pe nodul *worker-1*, unde sunt dispuse matricile trimise, dar și cele agregate și primite drept răspuns de la nodul central.

```
Sent matrix: [[0.914, 0.086], [0.892, 0.108]], Response: {'aggregated_matrix': [[0.914, 0.086], [0.892, 0.108]], 'status': 'aggregated'}  
Sent matrix: [[0.406, 0.594], [0.501, 0.499]]  
Sent matrix: [[0.406, 0.594], [0.501, 0.499]], Response: {'aggregated_matrix': [[0.38843750000000005, 0.6115625], [0.5569375000000001, 0.4430625]],  
Sent matrix: [[0.495, 0.505], [0.219, 0.781]]
```

Figura 12. Log-urile de pe o instanță de worker

Folosirea unor metrice de calculare a scorului preciziei prezicerii nu a avut însă sens, deoarece datele folosite sunt date de intrare generate în mod aleator, dar studiul din cadrul *proof of concept*-ului a dus la găsirea unui set de date care pare promițător și poate reprezenta un etalon atunci când vine vorba de testări ulterioare ale algoritmilor de predicție realizați.

## 7 CONCLUZII

În concluzie, putem spune că am demonstrat viabilitatea folosirii lanțurilor Markov pentru prezicerea mobilității umane la exterior, oferind o soluție relativ simplă comparativ cu metodele mai complexe de *Machine Learning*. Metodologia propusă, structurată pe etape bine definite, a permis realizarea unui *proof of concept*, care a validat topologia și funcționalitatea sistemului printr-un experiment local folosind *containere* Docker și framework-ul Flask.

*Proof of concept*-ul a implicat crearea unui prototip de topologie sumară, bazat pe un număr mic de entități comunicante, care au transmis datele acumulate către un server central pentru a realiza un model de predicție. Alegerea Docker și Flask ca tehnologii principale s-a dovedit a fi benefică datorită capacităților de virtualizare și flexibilității oferite pentru dezvoltarea rapidă a unui REST API necesar pentru agregarea și prelucrarea datelor.

Totuși, studiul a identificat și o serie de limitări ale metodologiei aplicate. Utilizarea unui număr redus de containere Docker nu a putut simula pe deplin complexitatea și dinamismul unui sistem real de prezicere a mobilității umane la scară largă. Creșterea numărului de noduri ar putea genera overhead și pierderi de pachete în comunicație, aspect care necesită optimizări suplimentare, cum ar fi utilizarea unei topologii descentralizate.

De asemenea, folosirea unor date aleatoare pentru a simula agregarea matricelor de tranziție nu este neapărat o metodă benefică de validare a soluției folosite, dar a pus

bazele unui studiu ulterior care implică, ca prim pas, studierea unui set de date care poate fi de ajutor în contextul evaluării modelului creat.

Deși lanțurile Markov au demonstrat rezultate promițătoare în acest context, literatura de specialitate sugerează existența unor algoritmi mai performanți bazați pe Machine Learning, care pot integra multiple variabile și factori externi, precum condițiile meteo și evenimentele sociale. Aceste metode avansate ar putea oferi o acuratețe sporită în prezicerea mobilității, adresând complexitatea și variabilitatea comportamentului uman în mediul exterior.

În concluzie, acest studiu a pus bazele unei soluții funcționale și a deschis calea pentru cercetări ulterioare și optimizări care să includă metode mai avansate și complexe, necesare pentru a obține o prezicere precisă și fiabilă a mobilității umane.

## 8 BIBLIOGRAFIE

- [1] European Commission. What is personal data?. [https://commission.europa.eu/law/law-topic/data-protection/reform/what-personal-data\\_en](https://commission.europa.eu/law/law-topic/data-protection/reform/what-personal-data_en). Ultima accesare: 04.02.2024
- [2] Regulation (EU) 2016/679 of the European Parliament and of the Council. General Data Protection Regulation. <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN#d1e1404-1-1>. Ultima accesare: 04.02.2024
- [3] Li, Tian & Sahu, Anit & Talwalkar, Ameet & Smith, Virginia. (2019). Federated Learning: Challenges, Methods, and Future Directions.
- [4] Juwon Park, Daegun Yoon, Sangho Yeo, Sangyoon Oh. (2022). AMBLE: Adjusting mini-batch and local epoch for federated learning with heterogeneous devices, Journal of Parallel and Distributed Computing, Volume 170, <https://doi.org/10.1016/j.jpdc.2022.07.009>.
- [5] Caldas, S., Konečný, J., McMahan, H.B., & Talwalkar, A. (2018). Expanding the Reach of Federated Learning by Reducing Client Resource Requirements. ArXiv, abs/1812.07210.
- [6] Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., & Bacon, D. (2016). Federated Learning: Strategies for Improving Communication Efficiency. ArXiv, abs/1610.05492.
- [7] He, Lie & Bian, Yatao An & Jaggi, Martin. (2018). COLA: Decentralized Linear Learning.
- [8] B. Recht, C. Re, S. Wright, and F. Niu, "HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent," in Proc. Advances in Neural Information Processing Systems, 2011, pp. 693–701.
- [9] Nishio, Takayuki & Yonetani, Ryo. (2019). Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. 1-7. 10.1109/ICC.2019.8761315.

- [10] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in Proc. Advances in Neural Information Processing Systems, 2017, pp. 4424–4434.
- [11] Wang, Jinzhong & Kong, Xiangjie & Xia, Feng & Sun, Lijun. (2019). Urban Human Mobility: Data-Driven Modeling and Prediction. ACM SIGKDD Explorations Newsletter. 21. 1-19. 10.1145/3331651.3331653.
- [12] Yan, Xiao-Yong & Wang, Wen-Xu & Gao, Zi-You & Lai, Ying-Cheng. (2017). Universal model of individual and population mobility on diverse spatial scales. Nature Communications. 8. 10.1038/s41467-017-01892-8.
- [13] Alam, Muhammad Raisul & Reaz, Mamun Bin Ibne & Mohd Ali, Mohd. (2012). SPEED: An Inhabitant Activity Prediction Algorithm for Smart Homes. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on. 42. 985-990. 10.1109/TSMCA.2011.2173568.
- [14] X. Song, H. Kanasugi, and R. Shibasaki. Deeptransport: prediction and simulation of human mobility and transportation mode at a citywide level. In International Joint Conference on Artificial Intelligence, pages 2618–2624, New York, 2016.
- [15] Geeks for Geeks. K means Clustering - Introduction. <https://www.geeksforgeeks.org/k-means-clustering-introduction>. Ultima accesare: 26.05.2024
- [16] Fu, Guoyu (Michael), Khider, Mohammed, van Diggelen, Frank, "Android Raw GNSS Measurement Datasets for Precise Positioning," *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*, September 2020, pp. 1925-1937. <https://doi.org/10.33012/2020.17628>
- [17] Kaggle. Let's visualize dataset to understand. <https://www.kaggle.com/code/nayuts/let-s-visualize-dataset-to-understand/notebook>. Ultima accesare: 26.05.2024
- [18] Docker. <https://www.docker.com/>. Ultima accesare: 26.05.2024
- [19] Prakhar Srivastav. Learn to build and deploy your distributed applications easily to the cloud with Docker. <https://docker-curriculum.com/>. Ultima accesare: 26.05.2024
- [20] Flask. <https://flask.palletsprojects.com/en/2.3.x/>. Ultima accesare: 26.05.2024.
- [21] Martijn Faassen. What is Pythonic?. <https://blog.startifact.com/posts/older/what-is-pythonic.html>. Ultima accesare: 26.05.2024.
- [22] Matt Makai. Full Stack Python. <https://www.fullstackpython.com/flask.html>. Ultima accesare: 26.05.2024.
- [23] Qiao, Yuanyuan & Si, Zhongwei & Zhang, Yanting & Ben Abdesslem, Fehmi & Zhang, Xinyu & Yang, Jie. (2017). A Hybrid Markov-based Model for Human Mobility Prediction. Neurocomputing. 278. 10.1016/j.neucom.2017.05.101.