



UNIVERSITATEA DIN
BUCUREŞTI



FACULTATEA DE
MATEMATICĂ ŞI
INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

UN PAS SPRE ÎNTELEGEREA ALFABETULUI ASL

Absolvent
Stratulat Vladimir

Coordonator științific
Conf. Dr. Alexe Bogdan

Bucureşti, iunie 2025

Rezumat

Scopul acestei lucrări este facilitarea interacțiunii cu persoanele care prezintă deficiențe de auz și/sau vorbire prin eliminarea barierelor de comunicare prezente între acestea și indivizii din populația generală.

Soluția propusă constă într-o aplicație mobilă capabilă să detecteze și să traducă semnele alfabetului *American Sign Language* (ASL). Nucleul aplicației este reprezentat de o rețea neuronală convezională, integrată într-o aplicație dezvoltată pentru sistemul de operare Android. Modelul urmează o arhitectură proprie, inspirată din modele performante existente, și este antrenat pe un set de date construit din multiple surse.

În procesul de antrenare au fost utilizate tehnici moderne de regularizare, precum *dropout*, *label smoothing* și augmentarea datelor, cu scopul îmbunătățirii generalizării modelului. Acuratețea finală pe un set de testare reprezentativ pentru mediul real este de peste 91%.

Se poate concluziona că lucrarea și-a atins obiectivul, modelul propus fiind performant și eficient, iar aplicația oferă o bază pentru extinderea cercetării către recunoașterea cuvintelor și a propozițiilor.

Abstract

The aim of this thesis is to bridge the gap between individuals with hearing and/or speech impairments and the general population.

The proposed solution consists of a mobile application capable of detecting and translating the American Sign Language (ASL) alphabet. At its core, the application is composed of a convolutional neural network, integrated into an Android mobile application. The model follows a custom architecture, inspired by state-of-the-art models, and was trained on a custom dataset composed of multiple sources.

During the training process, several modern regularization techniques were applied, such as dropout, label smoothing and data augmentation, with the purpose of improving model generalization. The final model achieved an accuracy of over 91% on a test set representative of real-world conditions.

It can be concluded that the thesis reached its goal, the proposed model being accurate and efficient. Moreover, the final application offers a foundation for extending the research towards recognizing words and sentences.

Cuprins

| | | |
|----------|--|-----------|
| 1 | Introducere | 6 |
| 2 | Recunoașterea alfabetului ASL | 10 |
| 2.1 | Construirea setului de date | 10 |
| 2.2 | Preprocesarea imaginilor | 11 |
| 2.3 | Arhitectura modelului | 13 |
| 2.4 | Antrenarea modelului | 15 |
| 2.5 | Evaluarea modelului | 16 |
| 2.6 | Evaluare experimentală | 19 |
| 3 | Aplicația mobilă și infrastructura aplicației | 20 |
| 3.1 | Aplicația Android | 20 |
| 3.2 | Infrastructura aplicației | 24 |
| 4 | Direcții viitoare și concluzii | 26 |
| 4.1 | Direcții viitoare | 26 |
| 4.2 | Concluzii | 27 |
| | Bibliografie | 28 |

Listă de figuri

| | | |
|-----|---|----|
| 1.1 | Alfabetul ASL | 7 |
| 1.2 | Dialecte pentru litera P | 7 |
| 1.3 | Literele din setul de date. | 8 |
| 2.1 | Aplicarea MediaPipe Hands | 10 |
| 2.2 | Distribuția claselor | 11 |
| 2.3 | Efecte intra-imagine | 12 |
| 2.4 | CutMix și MixUp | 13 |
| 2.5 | Arhitectura VladimirNet | 14 |
| 2.6 | Vizualizarea antrenării | 17 |
| 2.7 | Matricea de confuzie a setului de testare | 17 |
| 2.8 | Acuratețea pe clasă | 18 |
| 3.1 | Ecranele inițiale ale aplicației | 21 |
| 3.2 | Crearea unui cont nou | 22 |
| 3.3 | Conectare și resetarea parolei | 22 |
| 3.4 | Meniul principal | 23 |
| 3.5 | Sesiune de învățare | 23 |

Listă de tabele

| | |
|-------------------------------------|----|
| 2.1 Raport de clasificare | 18 |
|-------------------------------------|----|

Capitolul 1

Introducere

Un subdomeniu al inteligenței artificiale care a captat interesul publicului, al sectorului privat și al instituțiilor guvernamentale este vederea artificială. În ultimii ani au fost înregistrate progrese semnificative atât în dezvoltarea de noi metode utilizate în vederea artificială, cât și al capacitații de procesare a datelor și implementare a acestor noi algoritmi. Vederea artificială are un rol bine stabilit într-o gamă variată de cazuri, cum ar fi robotica, securitatea unei instituții, industria *software* (recunoaștere facială, realitate augmentată) și, cel mai important pentru lucrarea de față, medicina și accesibilitatea.

Această lucrare își propune să contribuie la apropierea de persoanele care prezintă deficiențe de auz și/sau vorbire. Având în vedere importanța integrării acestora și nevoia de a ușura interacțiunea, vizăm să reducem bariera apărută în comunicare. Aplicația dezvoltată sprijină indivizii care doresc fie să traducă literele exprimate cu ajutorul mâinilor, fie să învețe sau să exerseze alfabetul limbajului american al semnelor (în engl. *American Sign Language - ASL*).

Pentru a ne atinge scopul, utilizăm tehnici de *deep learning* pentru antrenarea unui model bazat pe o rețea neuronală conlovțională (în engl. Convolutional Neural Network - CNN), reprezentând nucleul aplicației mobile Android folosite de utilizator. Aplicația reprezintă interfața prin care utilizatorul interacționează cu rețeaua neuronală antrenată. Aceasta transmite modelului cadrele capturate cu ajutorul camerei dispozitivului, modelul le analizează și returnează rezultatele detecției.

În căutarea unui set de date potrivit pentru antrenare, am observat că majoritatea seturilor de date privind alfabetul ASL conțineau doar imagini similare, fie ca fundal, fie ca poziție a mâinii, făcând dificilă antrenarea unui model asupra unui singur set de date. Pentru a depăși această problemă, s-a optat pentru combinarea mai multor surse, pentru a ajuta modelul în generalizare. Un set de date a fost creat special pentru acest studiu prin înregistrarea video a autorului lucrării și a unui voluntar, trei seturi au fost preluate de pe **Youtube** [52, 42, 46], cinci au fost preluate de pe **Kaggle** [31, 22, 50, 23, 51], iar restul de cinci din articole științifice [30, 43, 20, 34, 38], integrând un total de 14 surse.

Pentru început, am remarcat faptul că seturile de date originale conțineau imagini cu

litere aparținând mai multor dialecte ale alfabetului ASL. Prin urmare, materialul oferit de *American Society for Deaf Children* (ASDC), ilustrat în Figura 1.1, servește drept referință în filtrarea manuală a imaginilor.

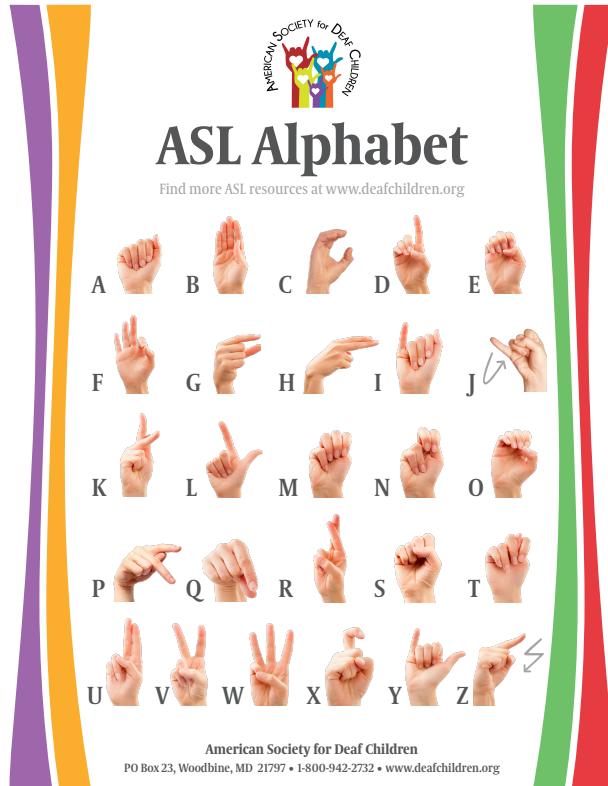


Figura 1.1: **Alfabetul ASL.** Imagine preluată de pe website-ul ASDC [1]. Fiecare palmă reprezintă o literă din alfabet. J și Z presupun mișcare, aşadar săgețile indică forma mișcării.

În cazul în care unul sau mai multe dialecte diferite de cel ilustrat în Figura 1.1 aveau reprezentativitate consistentă, s-a decis păstrarea acestora, încrucât modelul este capabil să învețe diferite dialecte pentru o literă, cât timp „vede” destule exemple. Un caz de acest fel poate fi observat în Figura 1.2.

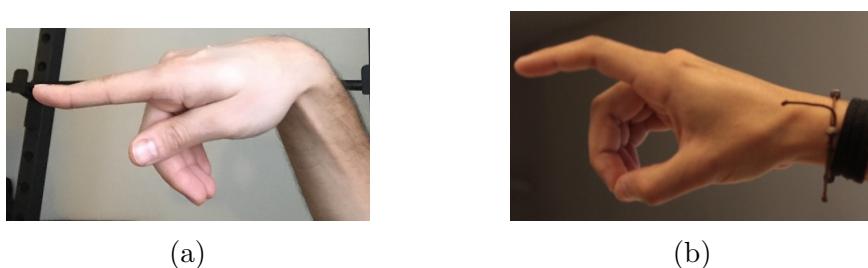


Figura 1.2: **Dialecte pentru litera P.** Ambele imagini ilustrează aceeași literă, P, și exemplifică două dialecte diferite, provenind din surse distincte.

Literele J și Z au fost eliminate deoarece presupun mișcarea mâinii, iar aplicația noastră analizează un singur cadru independent.

Câte un exemplu sugestiv pentru fiecare literă din setul de date final poate fi analizat în Figura 1.3.

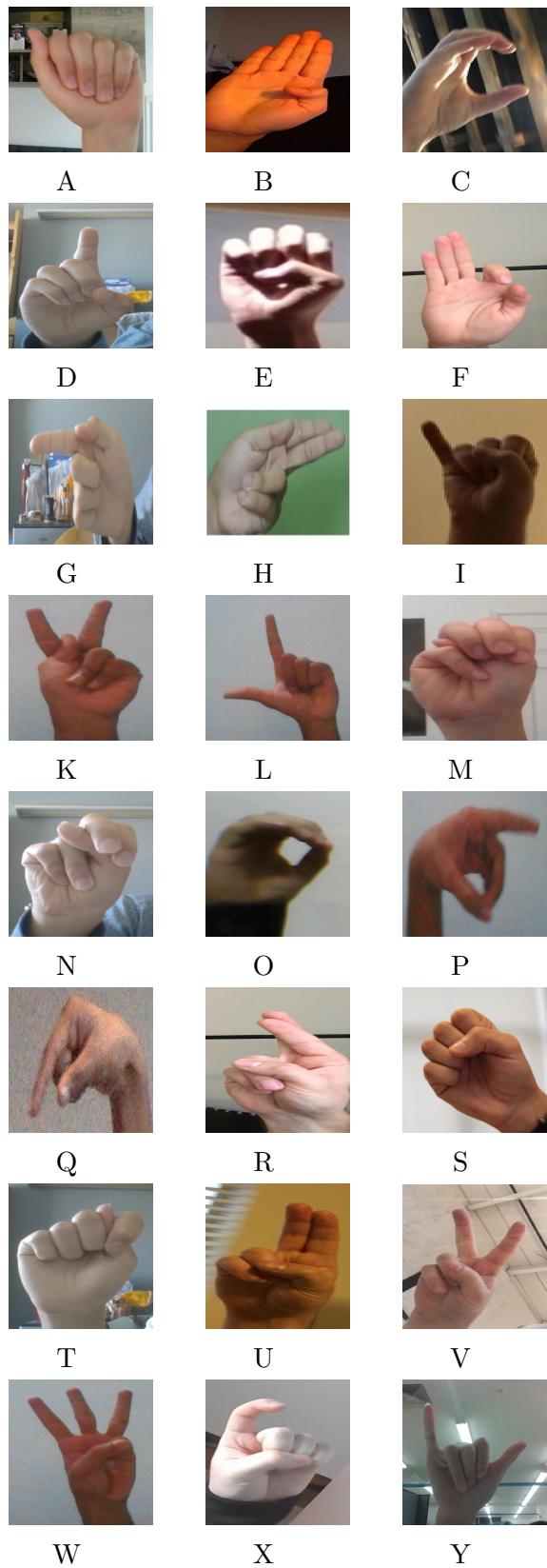


Figura 1.3: **Literele din setul de date.** Ilustrăm câte un exemplu aleator, pentru fiecare clasă. Literă situată sub fiecare imagine reprezintă clasa corespunzătoare.

Lucrarea de față este structurată în trei capitole principale, după cum urmează:

1. **Recunoașterea alfabetului ASL** - în acest capitol explicăm procesul care a stat la baza creării setului de date, preprocesarea imaginilor, arhitectura modelului și procesul de antrenare al acestuia. În final, oferim o evaluare experimentală în care analizăm arhitecturile și strategiile nereușite care au contribuit la construirea arhitecturii finale.
2. **Aplicația mobilă și infrastructura aplicației** - în acest capitol prezentăm ecranele principale ale aplicației Android, continuând cu o analiză a logicii aplicației și a capacitatei serverului.
3. **Direcții viitoare și concluzii** - în acest ultim capitol recapitulăm procesul științific, analizăm limitările și lipsurile aplicației, posibilele soluții privind limitele menționate și oferim direcții viitoare de dezvoltare.

Capitolul 2

Recunoașterea alfabetului ASL

2.1 Construirea setului de date

Extragerea cadrelor din videoclipuri. În cazul seturilor de date formate din clipuri video, a fost utilizat un script **Python** [35], care extrage și salvează cadrele citite cu ajutorul bibliotecii **OpenCV** [3], utilă în procesarea și analiza imaginilor.

Extragerea palmei din imagini. Pentru a extrage mâna, a fost utilizat algoritmul **MediaPipe Hands**, dezvoltat de **Google** [55] pentru detectia palmei într-o imagine. Acest pas a fost necesar pentru a elmina zgomotul creat de fundal sau de fețele oamenilor și pentru a aduce palma în prim plan. Transformarea unei imagini în urma aplicării algoritmului poate fi observată în Figura 2.1.

În cazul cadrelor extrase din videoclipuri și al palmelor extrase din imagini, imaginile invalide (neclaritate, poziție greșită a mâinii, mâna incorectă etc.) au fost eliminate manual.

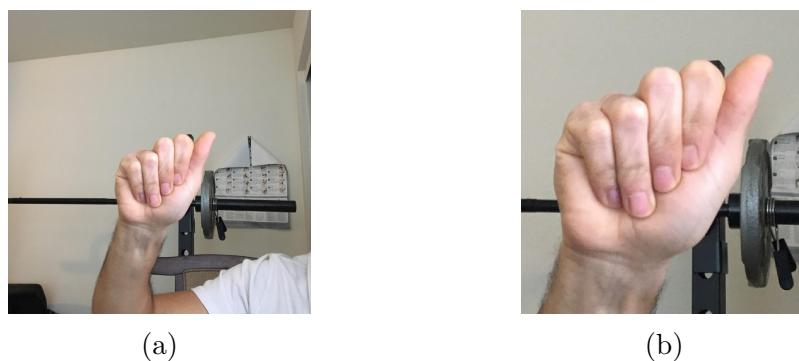


Figura 2.1: **Aplicarea MediaPipe Hands.** (a) prezintă imaginea originală care servește drept intrare pentru MediaPipe Hands; (b) este imaginea utilizată în antrenare, în urma extragerii acesteia. Se poate observa cum palma devine obiectul principal al imaginii.

Împărțirea setului de date. Pentru a ne asigura că acuratețea obținută pe seturile de validare și testare reflectă performanța în lumea reală, imaginile sunt împărțite manual, în funcție de sursa lor, astfel încât în seturile de validare și testare să nu existe imagini

cu mâini sau fundaluri aflate în setul de antrenare. În final, avem un total de 36.871 de imagini, împărțite după cum urmează: pentru setul de antrenare sunt alocate 30.450 de imagini, reprezentând 82,58% din total, pentru setul de validare 3.211 de imagini, reprezentând 8,7% din total, iar pentru setul de testare 3.210 de imagini, reprezentând 8,7% din total.

Distribuția claselor. În cazul setului de antrenare, literele subrepräsentate sunt M, N și P. Cauza acestei subrepräsentări este dificultatea pe care am întâlnit-o în colectarea unui număr suficient de exemple pentru un singur dialect. Cu toate acestea, un număr de aproximativ 1.000 de exemple pentru o clasă este considerat a fi suficient pentru ca modelul să poată învăța trăsăturile importante.

În cazul seturilor de validare și testare, distribuția este aproximativ egală, deoarece clasele nu au necesitat un număr ridicat de exemple, spre deosebire de cele care formează setul de antrenare. Distribuția exactă a seturilor poate fi observată în Figura 2.2.

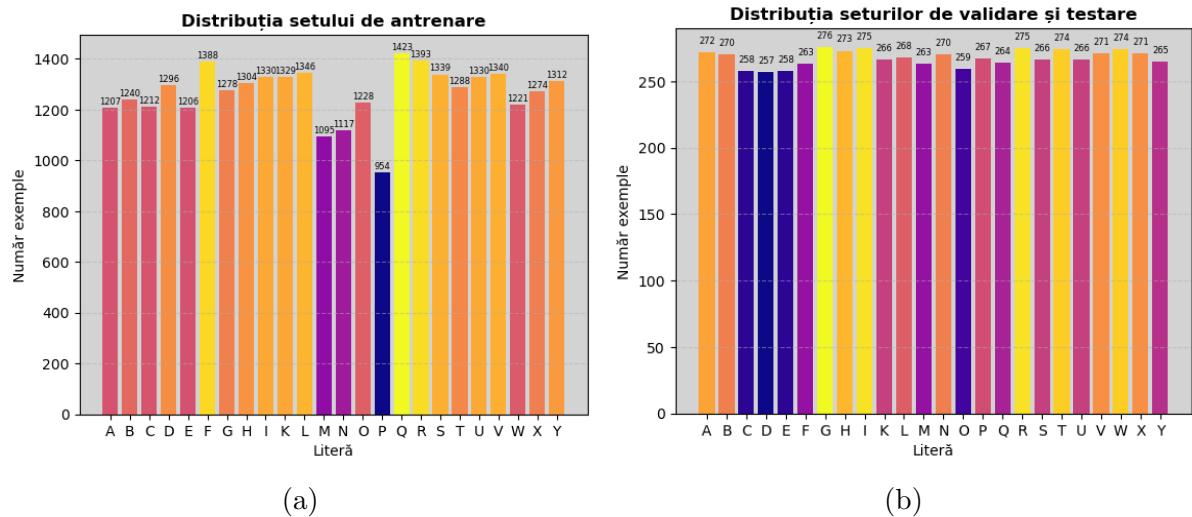


Figura 2.2: **Distribuția claselor.** (a) ilustrează distribuția setului de antrenare; (b) ilustrează distribuția setului care este împărțit aleator ($seed = 42$), în două seturi: validare și testare.

2.2 Preprocesarea imaginilor

Preprocesarea setului de antrenare. Cu scopul de a avea exemple cât mai variate, în special pentru a combate probleme ca *overfitting*-ul, am decis să utilizăm o tehnică numită augmentarea datelor. Efectele clasice (intra-imagine), care acționează independent asupra unei singure imagini, precum răsturnarea, rotirea, aplicarea de estompare sau zgromot, modificarea culorilor sau ștergerea unei porțiuni dintr-o imagine, pot îmbunătăți capacitatea de generalizare a unui model deep, cum ar fi un CNN [40, 26]. Pe lângă efectele intra-imagine, există și efecte care amestecă două imagini, efecte inter-imagine, cum ar fi *CutMix* [54] și *MixUp* [56].

Pentru efectele clasice, augmentările au fost realizate utilizând biblioteca **Albumentations** [4]. Augmentările aplicate sunt: răsturnare orizontală cu probabilitatea aplicării de 50%, rotire între -15° și 15° cu o probabilitate de 40%, estompare cu filtru de dimensiune 3x3 și probabilitate de 30%, zgomot Gaussian cu abatere standard între 0.05 și 0.07 și o probabilitate de 30% și trepidație de culoare sau *color jitter* (schimbarea luminozității, contrastului, saturăției și nuantei), cu o probabilitate de 40%. Toate imaginile sunt redimensionate la 224×224 , normalize folosind media per canal RGB [0.5797, 0.5104, 0.4846] și abatere standard [0.1804, 0.1845, 0.1883], reprezentând media și abaterea standard a setului de antrenare, și în final, transformate în tensori.

După cum se observă în Figura 2.3, efectele aplicate pot ajuta modelul să „înțeleagă” că factori precum culoarea pielii sau poziția exactă a mâinii nu sunt importanți, și mai mult de atât, să se descurce în condiții de luminozitate scăzută și cu imagini de calitate redusă.

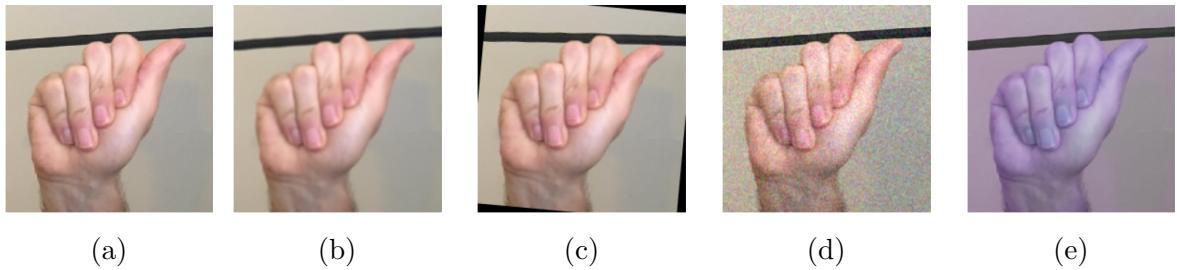


Figura 2.3: **Efecte intra-imagine.** Ilustrăm efectele aplicate asupra imaginilor, pentru augmentarea setului de antrenare. (a) exemplifică imaginea originală, (b) exemplifică efectul de estompare, (c) exemplifică rotirea, (d) exemplifică zgomotul adăugat, iar (e) exemplifică efectul de *color jitter*.

În cazul efectelor inter-imagine CutMix și MixUp am utilizat biblioteca **torchvision** [25]. La fiecare încărcare a unui *batch* de date din setul de antrenare, cu ajutorul unui obiect de tip *DataLoader* din biblioteca **PyTorch** [32], alegem aleatoriu dintre CutMix sau MixUp, cu o probabilitate de aplicare de 50%.

CutMix este definit ca:

$$\begin{aligned}\tilde{x} &= M \odot x_B + (1 - M) \odot x_A \\ \tilde{y} &= \lambda y_A + (1 - \lambda) y_B\end{aligned}\tag{2.1}$$

unde \tilde{x} este imaginea rezultată, x_B și x_A sunt imaginile sursă care vor fi combinate, iar M este masca binară care va fi înmulțită element cu element (\odot) cu imaginile sursă. \tilde{y} este eticheta *soft*, iar λ reprezintă ponderea fiecărei imagini în imaginea finală.

MixUp este definit ca:

$$\begin{aligned}\tilde{x} &= \lambda x_A + (1 - \lambda) x_B \\ \tilde{y} &= \lambda y_A + (1 - \lambda) y_B\end{aligned}\tag{2.2}$$

unde \tilde{x} este imaginea rezultată, x_A și x_B sunt imaginile sursă care vor fi combinate, iar λ este un coeficient ales aleator, care controlează proporția fiecărei imagini. \tilde{y} este la fel ca și în cazul ecuației pentru CutMix.

Este important de menționat că, în cazul nostru, păstrăm din tehnica CutMix și MixUp doar combinarea imaginilor, iar eticheta atribuită corespunde clasei dominante din imaginea compusă.

Pentru a vizualiza efectele aplicate asupra a două imagini, vezi Figura 2.4.

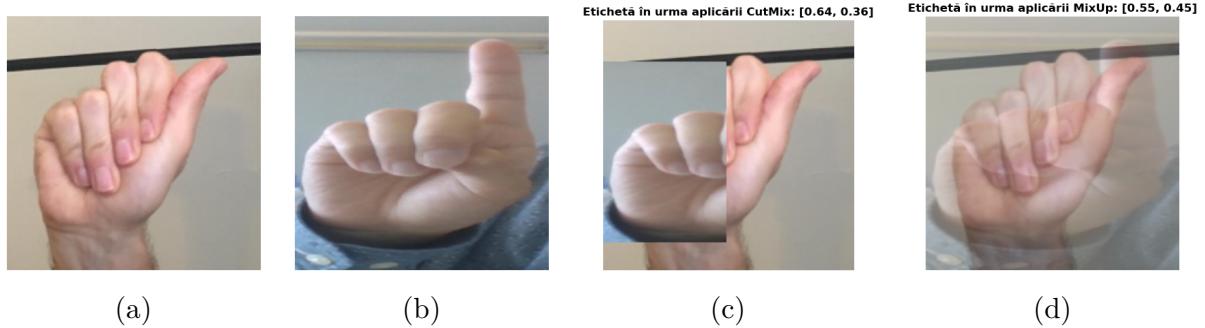


Figura 2.4: **CutMix și MixUp.** Ilustrăm efectele CutMix și MixUp. (a) și (b) reprezintă două exemple din două clase distincte; (c) exemplifică efectul CutMix, unde clasa din (a) are o pondere de 0.64, iar (b) are o pondere de 0.36; (d) exemplifică efectul MixUp, unde clasa (a) are o pondere de 0.55, iar clasa (b) are o pondere de 0.45.

Preprocesarea seturilor de validare și testare. În cazul acestor două seturi, imaginea originală este redimensionată la 224×224 , normalizată cu media și abaterea standard a setului de antrenare și transformată în tensori.

2.3 Arhitectura modelului

Arhitectura creată pentru acest studiu a fost inspirată din mai multe arhitecturi revoluționare, precum VGG [41] de unde a fost preluată ideea utilizării de bloc (mai multe straturi) convecțional, urmat de un strat de *max pooling*, tehnică care reduce dimensiunea hărților de activare.

În plus, arhitectura a fost inspirată de ResNet [11], prin inserarea unor straturi convecționale succesive care păstrează numărul de canale, consolidând și rafinând trăsăturile extrase în ultima extindere. Din aceeași arhitectură, a fost inspirată și inserarea straturilor de *batch normalization* după straturile convecționale, cu scopul de a acceleră procesul de învățare, permitând utilizarea unei rate de învățare mai mare, și de a oferi o ușoară regularizare prin reducerea variațiilor mari [16]. În final, a fost preluată și tehnica de *adaptive average pooling*, prin care este redusă înălțimea și lățimea imaginii la 1×1 , păstrând adâncimea (numărul canalelor).

Modelul este format din 12 straturi convecționale, fiecare urmat de câte un strat de *batch normalization* și activat cu ajutorul funcției de activare ReLU, fiind una dintre cele

mai utilizate funcții de activare în cazul modelelor de tip CNN, deoarece oferă acuratețe ridicată în sarcini precum clasificarea imaginilor [7, 28]. În final, este prezent un clasificator format din două straturi complet conectate (dense). Straturile conoluționale sunt împărțite în cinci blocuri care sunt explicate în continuare. Pentru o mai bună înțelegere a arhitecturii și a explicației ce urmează, vezi Figura 2.5.

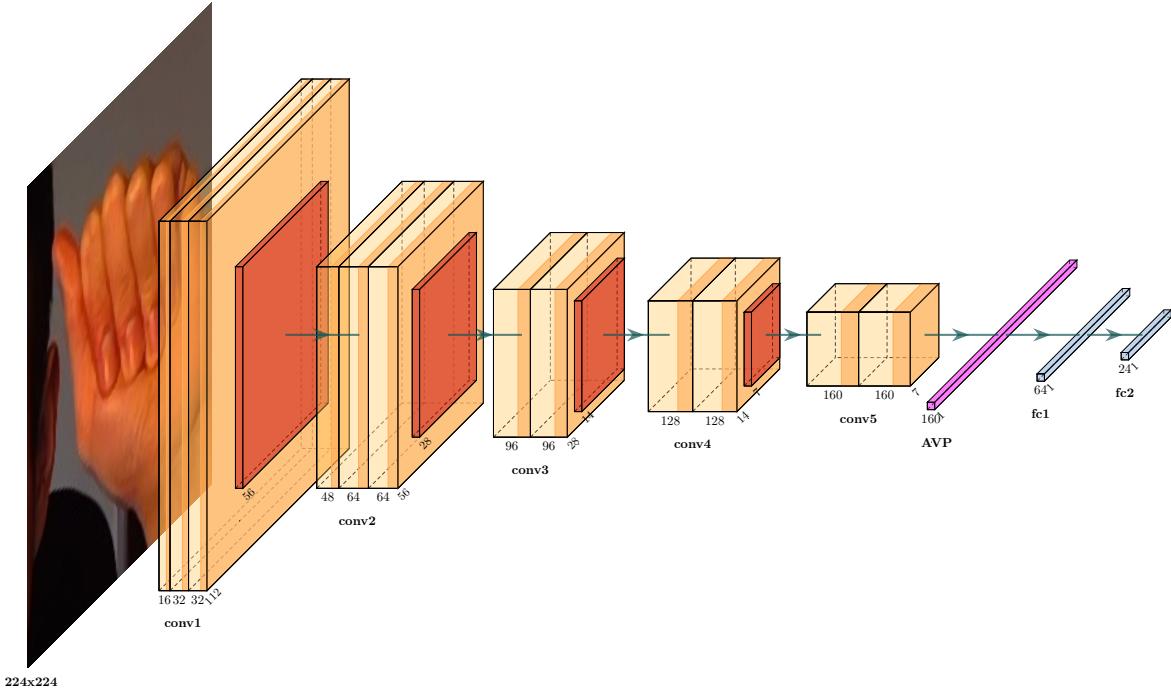


Figura 2.5: **Arhitectura VladimirNet**. Ilustrăm arhitectura modelului utilizat în lucrare, denumită **VladimirNet**. Conține un număr total de 927.496 de parametri. Prima imagine din stânga reprezintă imaginea care este transmisă modelului, cu dimensiunea initială de 224×224 . Blocurile conoluționale sunt notate cu conv1 , conv2 , conv3 , conv4 , conv5 . Fiecare strat este evidențiat cu galben transparent și are notat dedesubt numărul de canale în urma aplicării operației de conoluție. Straturile de batch normalization și ReLU sunt ilustrate cu portocaliu, iar culoarea roșie indică operația de max pooling. AVP reprezintă operația de adaptive average pooling. fc1 și fc2 sunt straturile dense ale clasificatorului. Diagrama a fost construită cu ajutorul bibliotecii *LATeX*, *PlotNeuralNet* [17].

- Primul bloc este format din trei straturi conoluționale. Primul strat extinde numărul canalelor de la trei (RGB) la 16, crescând adâncimea și micșorând imaginea la 112×112 , utilizând kernel de 5×5 și $\text{stride} = 2$ cu $\text{padding} = 2$. Dimensiunea este redusă la 112×112 pentru a nu consuma prea multă memorie la început, având în vedere dimensiunea kernel-ului de 5×5 care poate fi mai costisitor din punct de vedere computațional. Al doilea strat din blocul curent extinde din nou numărul de canale la 32 și utilizează tot un kernel de 5×5 , concentrându-se în continuare pe caracteristici generale. Ultimul strat, la fel ca toate straturile din finalul blocurilor conoluționale ale arhitecturii, nu adâncește imaginea, ci păstrează

numărul de canale din stratul anterior, însă de data aceasta folosind un kernel de 3×3 . Scopul ultimului strat este rafinarea și consolidarea trăsăturilor învățate pâna atunci, înainte de stratul de max pooling, cu rolul de a reduce dimensiunea spațială la jumătate. Fiecare bloc convoluțional din cele descrise în continuare este urmat de un strat de max pooling, cu excepția ultimului.

2. Al doilea bloc funcționează pe același principiu: 3 straturi convoluționale, ultimul păstrând numărul de canale, însă de acum înainte toate straturile utilizează un kernel cu dimensiunea de 3×3 .
3. Al treilea, al patrulea și al cincilea bloc conțin numai 2 straturi convoluționale, pentru a controla complexitatea modelului, având în vedere că ultimul strat atinge un număr de 160 de canale.
4. Ultimul bloc convoluțional nu mai este urmat de max pooling, ci de adaptive average pooling, strat care reduce dimensiunea spațială a fiecărei hărți la 1×1 , rezultând un vector cu 160 de valori.
5. În final, vectorul rezultat este transmis către clasificatorul format din două straturi complet conectate. Primul strat este urmat de ReLU și un strat de *dropout* cu $p = 0.45$, care dezactivează aleatoriu 45% dintre neuroni și conexiunile acestora, metodă necesară pentru a combate overfitting-ul [45, 29]. Ultimul strat dens este cel care ne oferă predicția.

De asemenea, este important să menționăm că ponderile sunt inițializate prin metoda *Kaiming*, care folosește valori dintr-o distribuție normală. Scopul acestei inițializări este de a evita gradienți care tind spre 0 (*vanishing gradient*) sau care „explosează” (*exploding gradient*) [12].

2.4 Antrenarea modelului

Antrenarea modelului a fost realizată în platforma **Google Colab** [8], folosind un **GPU NVIDIA L4**.

Funcția de pierdere. Pentru evaluarea predicțiilor oferite de către model, este utilizată entropia încrucișată (în engl. *Cross Entropy*) implementată cu ajutorul clasei *CrossEntropyLoss* din PyTorch, după formula:

$$L = - \sum_{i=1}^C y_i \cdot \log(\hat{y}_i) \quad (2.3)$$

unde C este numărul de clase posibile, y_i reprezintă vectorul *one-hot* a etichetei corecte, iar \hat{y}_i este vectorul rezultat din ultimul strat dens, asupra căruia este aplicată funcția de activare *softmax*.

Ca metodă de regularizare, vom folosi și o tehnică denumită *label smoothing* (netezirea etichetelor) [48], astfel încât y_i , vectorul one-hot *hard* utilizat de funcția cross entropy, este înlocuit cu unul soft care îndepărtează modelul de predicții excesiv de încrezătoare.

Algoritmul de optimizare. Pentru antrenarea modelului, s-a hotărât utilizarea algoritmului de optimizare *AdamW* [24], bazat pe Adam (Adaptive Moment Estimation) [21]. Îmbunătățirea adusă de către AdamW algoritmului de optimizare Adam constă în decuplarea penalizării ponderilor mari (în engl. *weight decay*) de calculul gradientului, astfel încât formula pentru actualizarea ponderilor este următoarea:

$$\theta_{t+1} = \theta_t - \eta \cdot \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \cdot \theta_t \right) \quad (2.4)$$

unde θ_t sunt ponderile în pasul t , η este rata de învățare, \hat{m}_t este media exponentială a gradientelor în pasul t , \hat{v}_t este media exponentială a pătratelor gradientelor în pasul t , iar λ reprezintă factorul de penalizare a greutăților.

Hiperparametri. Pentru optimizator, am ales o rată de învățare $\eta = 1 \times 10^{-3}$, aceasta fiind valoarea implicită din clasa *AdamW* a bibliotecii PyTorch. Pe parcursul învățării, η este ajustat treptat, la final având o valoare de 1×10^{-5} . Ajustarea este facilitată de clasa *CosineAnnealingLR* din PyTorch, care actualizează η la finalul fiecărui batch. Scopul acestei ajustări este de a scoate modelul din anumite minime locale nedorite, iar pe parcursul învățării, actualizările parametrilor să fie din ce în ce mai fine.

După cum am menționat mai devreme, etichetele sunt „netezite” înainte de a fi oferite funcției de pierdere, cu un factor de netezire $\epsilon = 0.15$, penalizând predicțiile excesiv de încrezătoare.

Timpul de antrenament este de 60 de epoci, cu mărimea batch-ului de 64 de exemple.

Pentru weight decay, utilizăm o valoare de 1×10^{-6} . Valoarea implicită pentru weight decay este de 1×10^{-2} , însă o valoare aşa ridicată conduce către o învățare mai lentă.

2.5 Evaluarea modelului

În timpul antrenării, la finalul fiecărei epoci, performanța modelului era evaluată pe setul de validare. Modelul a fost salvat în fiecare punct maxim al acurateții obținute pe setul de validare.

Evaluările au fost create și vizualizate cu ajutorul bibliotecilor **scikit-learn** [33], **matplotlib** [14] și **seaborn** [53].

Putem observa în Figura 2.6a evoluția modelului pe parcursul celor 60 de epoci. Acuratețea maximă pe setul de antrenare este de 92,45%, iar pe setul de validare de 92,27%. Datorită augmentărilor puternice și dificultății setului de antrenare, comparativ cu cel de validare, acuratețea setului de validare s-a menținut aproape constant deasupra acurateții setului de antrenare, iar spre final, acuratețea setului de antrenare ajungând să stagneze,

cu o ușoară tendință descendentală.

În Figura 2.6b putem observa cum funcția de pierdere scade treptat, indicând o învățare continuă.

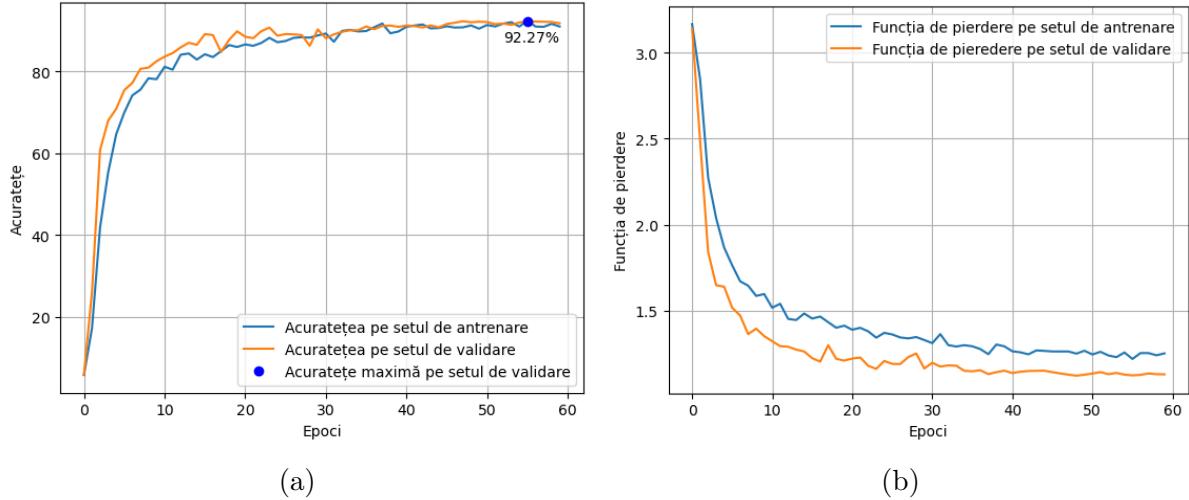


Figura 2.6: **Vizualizarea antrenării.** (a) ilustrează evoluția acurateții pe parcursul celor 60 de epoci de antrenare; (b) ilustrează evoluția funcției de pierdere pe parcursul antrenării.

După obținerea unor rezultate satisfăcătoare asupra testului de validare, modelul a fost evaluat și prin prisma setului de testare, obținând o acuratețe de 91,05%. În Figura 2.7 poate fi analizată matricea de confuzie obținută în urma evaluării rezultatelor setului de testare. Literele G și H sunt asemănătoare, la fel și literele N și M, motiv pentru care modelul le confundă mai des. Litera X este de asemenea confundată cu litera C, iar litera Y cu litera A.

| | | Matricea de confuzie a setului de testare | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|-----|---|-----|---|-----|-----|-----|-----|----|-----|-----|-----|-----|-----|-----|-----|-----|---|---|-----|-----|-----|-----|-----|---|---|----|-----|---|---|
| | | Adevăr | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | a | b | c | d | e | f | g | h | i | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | | | | | |
| a | 119 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15 | | | |
| b | 0 | 147 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | | | |
| c | 0 | 1 | 134 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| d | 0 | 0 | 0 | 0 | 114 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 5 | 9 | 0 | 4 | 0 | 0 | 0 | | | |
| e | 0 | 0 | 0 | 0 | 0 | 114 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| f | 0 | 4 | 0 | 2 | 0 | 0 | 104 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 0 | | | |
| g | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 108 | 13 | 0 | 0 | 0 | 1 | 3 | 1 | 9 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| h | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 117 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| i | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 142 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | | | |
| k | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 125 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | | | |
| l | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 135 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | |
| m | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 113 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 22 | 122 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| o | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 114 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| p | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 123 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| q | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 16 | 1 | 7 | 116 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| r | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 117 | 0 | 1 | 1 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | | | |
| s | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 134 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| t | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 5 | 2 | 0 | 1 | 0 | 2 | 114 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| u | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 1 | 0 | 122 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| v | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 127 | 2 | 0 | 0 | 0 | 0 | 0 | | |
| w | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 4 | 124 | 0 | 0 | 0 | 0 | 0 | 0 | |
| x | 0 | 0 | 0 | 0 | 23 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 109 | 0 | 0 | 0 | 0 | 0 | 0 |
| y | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 129 | 0 | 0 |

Figura 2.7: **Matricea de confuzie a setului de testare.** Ilustrăm matricea de confuzie cu ajutorul unei hărți termice. Diagonala principală reprezintă numărul de exemple prezise corect pentru fiecare literă.

Mai departe, în Figura 2.8 poate fi analizată acuratețea pentru fiecare literă. Literele cele mai ușor de recunoscut de către modelul creat sunt B și L, cu acuratețe de 99,3%, iar cele care ridică probleme sunt G, cu o acuratețe de 76,1%, și X, cu acuratețe de 80,1%.

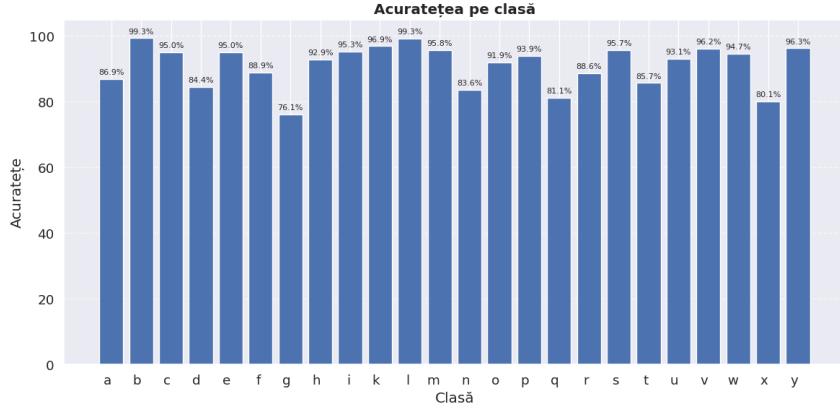


Figura 2.8: **Acuratețea pe clasă.** Ilustrăm acuratețea predicțiilor pentru fiecare literă. Acuratețea este notată în vârful dreptunghiurilor reprezentative pentru clasa respectivă.

În final, am creat un raport de clasificare. În Tabela 2.1 putem observa cum litera G are un scor *recall* scăzut (0.76), ceea ce înseamnă că există dificultăți în a recunoaște acea clasă. Combinând acest fapt cu datele din matricea de confuzie, putem concluziona că de multe ori, modelul are tendința să confundă G cu H, clasificând litera G ca fiind H. Mai mult de atât, chiar dacă litera Y are o acuratețe ridicată, poate fi observat cum scorul *precision* este mai scăzut (0.86), sugerând faptul că modelul tinde să prezică Y și în situații în care aceasta nu este litera corectă.

| Clasă | Precizie | Recall | Scor F1 | Număr exemple |
|-------|----------|--------|---------|---------------|
| a | 0.90 | 0.87 | 0.88 | 137 |
| b | 0.97 | 0.99 | 0.98 | 148 |
| c | 0.95 | 0.95 | 0.95 | 141 |
| d | 0.80 | 0.84 | 0.82 | 135 |
| e | 0.96 | 0.95 | 0.95 | 120 |
| f | 0.97 | 0.89 | 0.93 | 117 |
| g | 0.90 | 0.76 | 0.82 | 142 |
| h | 0.88 | 0.93 | 0.90 | 126 |
| i | 0.97 | 0.95 | 0.96 | 149 |
| k | 0.96 | 0.97 | 0.97 | 129 |
| l | 1.00 | 0.99 | 1.00 | 136 |
| m | 0.79 | 0.96 | 0.87 | 118 |
| n | 0.82 | 0.84 | 0.83 | 146 |
| o | 0.89 | 0.92 | 0.90 | 124 |
| p | 0.88 | 0.94 | 0.91 | 131 |
| q | 0.91 | 0.81 | 0.86 | 143 |
| r | 0.88 | 0.89 | 0.88 | 132 |
| s | 0.95 | 0.96 | 0.95 | 140 |
| t | 0.97 | 0.86 | 0.91 | 133 |
| u | 0.94 | 0.93 | 0.93 | 131 |
| v | 0.89 | 0.96 | 0.93 | 132 |
| w | 0.96 | 0.95 | 0.95 | 131 |
| x | 0.88 | 0.80 | 0.84 | 136 |
| y | 0.86 | 0.96 | 0.91 | 134 |

Tabela 2.1: Pentru fiecare clasă, raportăm numărul de exemple testate și scorurile: precizie, recall și F1.

2.6 Evaluare experimentală

Înainte de a avea arhitectura finală a rețelei neuronale conoluționale, au fost încercate diferite modele și tipuri de imagini.

În faza incipientă, imaginile au fost utilizate în forma lor originală, fără decuparea mâinii cu ajutorul MediaPipe Hands. Pentru a ne asigura că setul de date este corect, am antrenat un model cu arhitectura ResNet50 timp de 50 de epoci, obținând o acuratețe de 89,5% pe setul de validare. După ce am confirmat corectitudinea setului de date, s-a optat pentru arhitecturi mai simple, deoarece ResNet50 are aproximativ 25M parametri, ceea ce este considerat ineficient pentru rularea locală pe dispozitive mobile.

Încercând cu arhitecturi proprii, am ajuns la acuratețe de aproximativ 68%, cu modele care au între 300.000 și 10.000.000 de parametri. Principalele dificultăți întâmpinate au fost reprezentate de subînvățare și supraînvățare. Pe lângă numărul de parametri, arhitectura a ridicat probleme și în funcție de așezarea straturilor conoluționale și rata creșterii numărului de canale. În cazul în care dublam numărul canalelor la fiecare strat conoluțional, de exemplu $3 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 512$, modelul devine rapid prea complex și evaluarea începea să indice memorarea setului de date.

Printre algoritmii de ajustare a ratei de învățare, se numără *ReduceLROnPlateau*, care reduce η când învățarea stagniază, și *CosineAnnealingWarmRestarts*, care resetează η la un anumit punct, readucând rata de învățare la valoarea inițială, pentru a scăpa din minime locale și a încerca alte „drumuri” către convergență. În ambele cazuri, rezultatele nu erau cele așteptate, cele mai bune rezultate fiind observate prin utilizarea algoritmului CosineAnnealing.

Un punct de cotitură în cadrul testelor noastre a fost atins când am hotărât să utilizăm MediaPipe Hands pentru a extrage cadre cu mâinile. Acuratețea a început să crească considerabil, atingând deseori 90%, cu arhitecturi care conțineau aproximativ 1.000.000 de parametri.

Pe lângă modelul prezentat în Capitolul 2, am antrenat și un MobileNetV3Large [13], care a atins acuratețe de 95, 96% pe setul de validare. Scopul acestui model era de a-l introduce ca opțiune de utilizare în aplicație pentru dispozitive performante, însă după convertirea într-un format compatibil cu Android, timpul de detectie pe cadru creștea considerabil și nu am putut identifica cauza precisă.

În ceea ce privește augmentarea datelor, trebuie menționat că, în cazul CutMix și Mixup, etichetele trebuie reprezentate ca vectori soft, conform articolului original dedicat metodei CutMix. Autorii menționează că utilizarea doar a etichetei dominante oferă rezultate mai bune decât absența acestei metode de augmentare, însă rezultatele cele mai bune vor fi obținute prin utilizarea etichetelor soft. În urma experimentelor noastre, am observat că prin utilizarea etichetelor soft, a arhitecturii dezvoltate și a hiperparametrilor prezentați în această lucrare, acuratețea maximă obținută a fost de doar 89,15%.

Capitolul 3

Aplicația mobilă și infrastructura aplicației

3.1 Aplicația Android

Aplicația Android servește drept mediu de interacțiune cu modelul discutat în capitolul anterior. Aceasta este scrisă cu ajutorul limbajului **Kotlin** [18], utilizând biblioteca de instrumente pentru dezvoltare, **Jetpack Compose** [9]. Detecțiile au loc la nivel local, modelul fiind încărcat pe dispozitivul Android cu ajutorul **PyTorch Mobile**. Interfața pentru cameră este oferită de biblioteca **CameraX** [6], iar cererile *HTTP*, transmise prin protocolul securizat *HTTPS*, sunt facilitate de biblioteca **Retrofit** [44]. Ecranele prezentate în acest capitol funcționează pe baza unei stive gestionate cu ajutorul unui obiect de tip *NavController*. În continuare, detaliem structura și fluxul logic al aplicației.

Ecranul de bun-venit. După deschiderea aplicației, utilizatorul este întâmpinat de ecranul de bun-venit (Figura 3.1a), unde îi sunt prezentate trei opțiuni: *Sign In* (autentificare), *Sign Up* (crearea unui cont nou) sau *Skip and use offline* (simpla detectare a literelor). Aplicația afișează acest meniu doar în cazurile în care utilizatorul nu este autentificat sau este autentificat, dar dispozitivul nu este conectat la internet.

Utilizarea fără conexiune la internet. Această opțiune oferă posibilitatea de a detecta literele fără autentificare, pentru a asigura o experiență cât mai plăcută și accesibilă utilizatorilor care își doresc să acceseze doar funcția de traducere. Aplicația încarcă modelul în memorie și încearcă să detecteze mâini afișate pe ecran cu ajutorul algoritmului MediaPipe Hands. Odată ce o mână este detectată, este ocupat un dreptunghi care încadrează mâna și transmis mai departe către modelul responsabil pentru detectia literelor. În cazul în care este detectat un semn valid, este afișată litera recunoscută (Figura 3.1b), iar în caz contrar, este afișat caracterul „-” (Figura 3.1c).

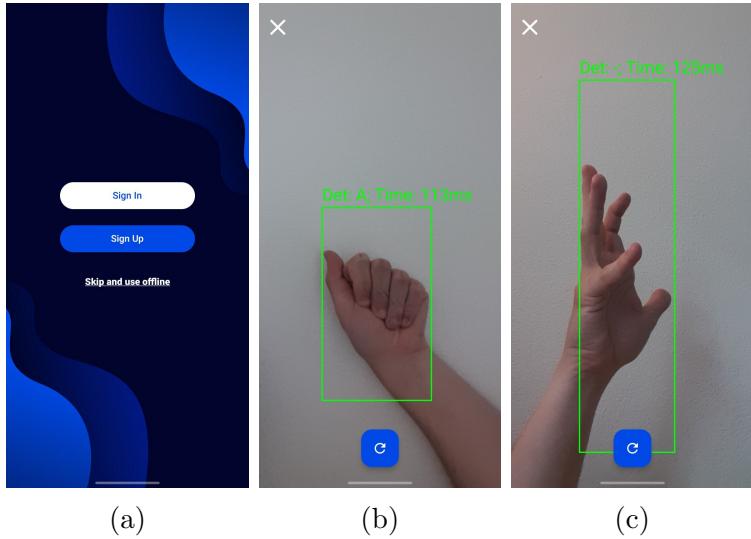


Figura 3.1: Ecranele inițiale ale aplicației. Ilustrăm ecranele inițiale ale aplicației. (a) ilustrează ecranul de bun-venit, unde sunt prezente butoanele de *Sign In*, *Sign Up* și *Skip and use offline*; (b) ilustrează modul de detectie în lipsa conexiunii la internet; palma detectată de MediaPipe Hands este încadrată în chenarul verde, deasupra căruia utilizatorul este înștiințat ca a fost detectată litera A, într-un timp de 113ms; (c) ilustrează momentul în care utilizatorul afișează un semn invalid.

Autentificarea utilizatorilor. Utilizatorul are posibilitatea de a crea un cont nou, de a se conecta cu datele sale, de a schimba parola și de a se deconecta.

Pentru început, utilizatorul apasă butonul de *Sign Up*, pentru a introduce adresa de email și parola noului cont (Figura 3.2a). După o validare a câmpurilor, aplicația trimite datele către server. În cazul în care serverul oferă un răspuns pozitiv, utilizatorului îi este afișat ecranul de validare a adresei de email (Figura 3.2b). În acest pas, utilizatorul află că are timp 5 minute pentru a introduce un cod primit pe email cu scopul de a confirma adresa. În urma unui răspuns valid din partea serverului privind codul de validare introdus, utilizatorul este înștiințat și redirecționat către ecranul de start, de unde se poate conecta cu noul său cont.

În urma apăsării butonului de *Sign In*, este afișat un ecran care oferă posibilitatea de a introduce datele contului (Figura 3.3a), care sunt transmise către server pentru validare. În cazul unui răspuns pozitiv, utilizatorul este direcționat către meniul principal, unde poate opta pentru începerea unei sesiuni de antrenament în învățarea alfabetului ASL. Mai mult decât atât, odată cu răspunsul pozitiv, dispozitivul primește un **JSON Web Token** (JWT) [19], denumit token de acces, alături de un token de reîmprospătare. Detalii despre această metodă sunt oferite în Secțiunea 3.2.

În cazul în care utilizatorul își uită parola, acesta poate face o cerere de schimbare a parolei, care constă în introducerea adresei de email către care sunt trimise instrucțiunile pentru resetare (Figura 3.3b).

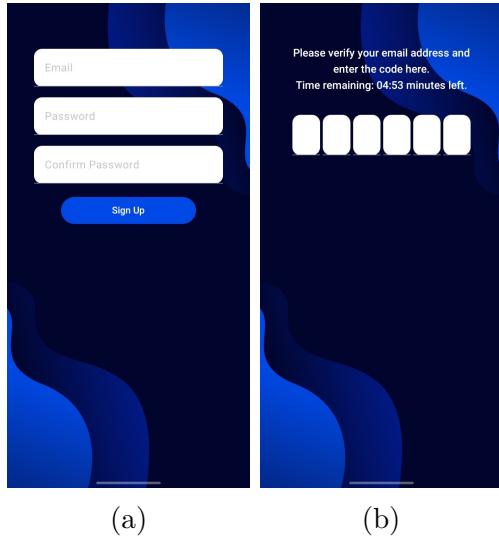


Figura 3.2: **Crearea unui cont nou.** Ilustrăm procesul de creare a unui cont nou. (a) ilustrează cele trei câmpuri necesare creării unui nou cont: email, parola, confirmarea parolei; (b) ilustrează înștiințarea utilizatorului de timpul rămas pentru introducerea codului de validare primit pe email, printr-un temporizator; în fiecare câmp trebuie introdusă câte o cifră.

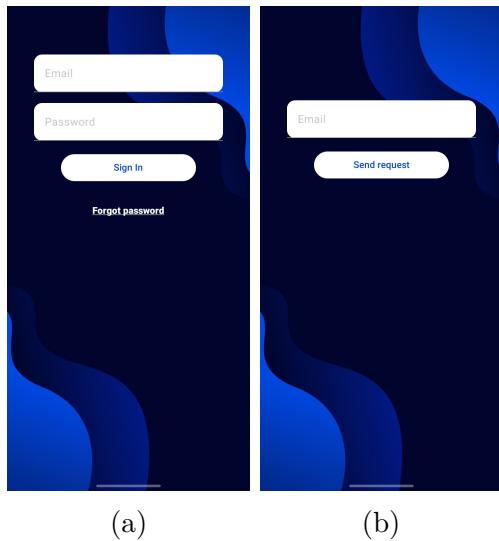


Figura 3.3: **Conectare și resetarea parolei.** (a) ilustrează câmpurile necesare pentru conectare (email și parola), alături de butoanele *Sign In* pentru conectare și *Forgot Password* pentru resetarea parolei; (b) ilustrează ecranul pentru crearea unei cereri de restare a parolei.

Meniul principal. Meniul prezentat în Figura 3.4 îi permite utilizatorului să aleagă între a învăța alfabetul ASL și a se deconecta. Deconectarea trimite o cerere către server pentru a invalida sesiunea și șterge token-urile prezente pe dispozitiv. Dacă utilizatorul apasă butonul *Start learning*, camera se deschide și este inițiată o nouă sesiune de antrenament.

Sesiune de învățare a alfabetului ASL. Odată cu începerea unei sesiuni de antrenament, utilizatorul primește înștiințarea că este necesară introducerea mâinii în cadru (Figura 3.5a). Pe urmă, este afișată prima literă din alfabet, care rămâne pe ecran cât timp modelul nu recunoaște corect litera indicată (Figura 3.5b). Odată ce litera afișată este detectată, sesiunea continuă automat la următoarea literă din alfabet (Figura 3.5c).

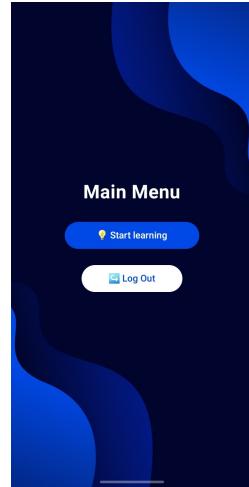


Figura 3.4: **Meniul principal.** Ilustrăm meniul principal care oferă opțiunile de începere a unei sesiuni de antrenament (Start Learning) și de deconectare (Log Out).

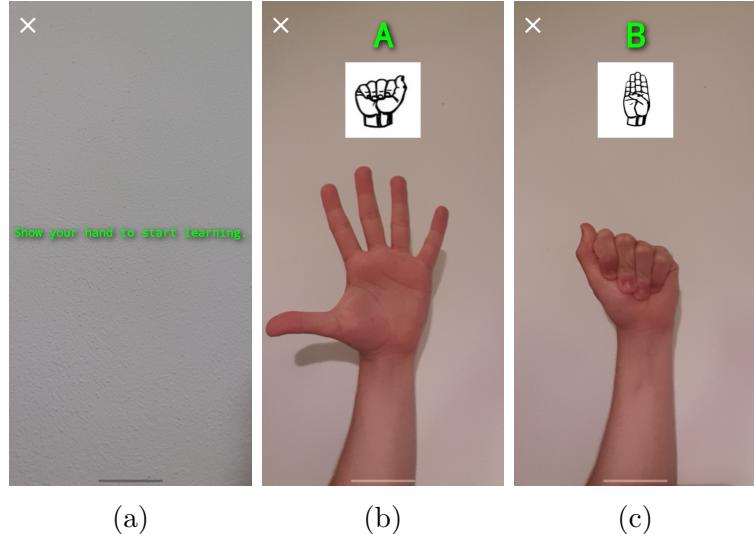


Figura 3.5: **Sesiune de învățare.** (a) ilustrează instrucțiunea primită de utilizator pentru a își introduce palma în cadru; (b) ilustrează începerea sesiunii de învățare prin afișarea literei A; (c) ilustrează afișarea literei B. Utilizatorul află ce literă învață prin afișarea caracterului cu culoarea verde, iar dedesubt se află o imagine care ilustrează semnul pentru acea literă.

3.2 Infrastructura aplicației

Arhitectura aplicației urmează o structură containerizată folosind **Docker** [27], prin *docker-compose*, ceea ce oferă un mediu de execuție portabil și izolat. Aplicația are la bază 4 containere:

- **Uvicorn + FastAPI** - server asincron [5] care răspunde cererilor HTTP prin cadrul de dezvoltare **FastAPI** [36], care gestionează logica aplicației și oferă suport în gestionarea măsurilor de securitate.
- **PostgreSQL** - bază de date relațională, care asigură stocarea persistentă a datelor [10].
- **Redis** - bază de date în memorie care funcționează pe bază de cheie-valoare [39] și asigură accesul rapid asupra datelor.
- **NGINX** - reverse proxy [47] care preia cererile HTTP și le redirecționează către serverul Uvicorn, gestionează traficul și asigură conexiuni securizate prin *SSL*.

Server. Serverul are la bază o arhitectură orientată pe servicii (SOA), structurată în straturi decuplate în funcție de responsabilitate. O cerere HTTP parcurge un flux logic de tipul: **API → serviciu** (logica aplicației) → **CRUD** (acces la baza de date) → răspuns.

Conexiunea către baza de date este creată utilizând componenta de acces la baza de date în mod asincron oferită de biblioteca **asyncpg** [15], iar relația dintre entitățile bazei de date și modelele relaționale definite în cod este facilitată de biblioteca **SQLModel** [37].

Relația de dependență dintre servicii este facilitată de clasa *Depends* din biblioteca FastAPI, care asigură mecanismul de injectare a dependențelor. De exemplu, clasa *AuthService* este injectată în *endpoint-urile* pentru autentificare, iar clasele *UserService* și *RedisClient* sunt injectate în clasa AuthService, aceasta din urmă depinzând de cele două.

Pe lângă înregistrarea de conturi noi și capacitatea de a conecta și deconecta utilizatori de la aplicație, serverul are ca funcționalități și schimbarea parolelor și reîmprospătarea token-urilor de acces.

Pentru procesarea unei cereri de înregistrare, este validată adresa de email cu ajutorul bibliotecii **email-validator** [49]. În urma validării, se verifică lipsa adresei în baza de date și a potrivirii textelor introduse în cele două câmpuri desemnate parolei. În continuare, se aplică o funcție de dispersie (în engl. *hash*) asupra parolei cu ajutorul bibliotecii **bcrypt** [2], iar datele utilizatorului sunt salvate. În final, este generat un cod unic de verificare care este trimis printr-un email către utilizator, cu scopul de a verifica adresa de email. Adresa și codul de verificare sunt salvate ca pereche cheie-valoare, unde codul reprezintă cheia și emailul reprezintă valoarea, într-o bază de date Redis, cu timp de expirare de

cinci minute. Utilizatorul nu poate folosi datele sale de acces până nu confirmă adresa de email.

În urma validării contului de email, utilizatorul se poate conecta la noul său cont, moment în care sunt verificate datele sale de autentificare. Alături de un răspuns pozitiv, serverul trimite și un obiect de tip *AuthorizationTokens*, care conține un token de acces și un token de reîmprospătare.

Pentru deconectare, clientul trimite către server token-ul de acces salvat pe dispozitiv și token-ul de reîmprospătare asociat. În cazul validării token-ului de acces, token-ul de reîmprospătare este invalidat, iar utilizatorul este deconectat și redirecționat către ecranul de bun-venit.

Token de acces și token de reîmprospătare. Timpul de expirare a token-ului de acces este, în cazul nostru, de o oră, și este semnat cu o cheie secretă, pentru a ne asigura, la recepționare, că acesta nu a fost manipulat în vreun fel. Avantajele JWT provin din principiul lipsei de stare pe care îl respectă, acesta nefiind stocat pe serverul aplicației, ci doar pe dispozitivul utilizatorului.

Rolul token-ului de reîmprospătare este vizibil în momentul în care expira JWT. Aceasta este stocat în baza de date Redis și este folosit pentru crearea unui nou JWT. Tandemul token de acces - token de reîmprospătare oferă utilizatorului o experiență plăcută în navigarea prin aplicație, deoarece toate verificările sunt făcute în fundal, fără a fi necesare reconectări manuale frecvente. Token-ul de reîmprospătare este valabil, în cazul nostru, timp de 30 de zile, perioadă la finalul căreia utilizatorul trebuie să se conecteze din nou.

Persistența datelor. Stocarea datelor este facilitată de o bază de date relațională de tip PostgreSQL și de o bază de date în memorie, Redis.

Baza de date relațională este folosită pentru a salva datele utilizatorilor și conține o tabelă numită *users*. Coloana *id* este cheia primară, de tip *UUID4*, *email* și *password* sunt de tip *VARCHAR*, iar *confirmed* este de tip *boolean* și asigură confirmarea contului.

Pentru a avea o experiență fluentă în utilizarea aplicației, am optat pentru stocarea token-urilor de reîmprospătare și a codurilor de acces într-o bază de date Redis. Tokenurile de reîmprospătare, codurile de validare a adreselor de email și cele pentru schimbarea parolelor sunt accesate cu un timp de latență minim. Toate cheile sunt salvate după ce au fost schimbată de o funcție hash, pentru a spori securitatea.

Reverse proxy. Toate cererile ajung mai întâi la containerul pentru NGINX, care le filtrează în funcție de cerințele notate în fișierul pentru configurații. În primul rând, sunt acceptate doar cereri HTTPS, containerul utilizând un certificat SSL. NGINX primește cereri la adresa <https://localhost:443>. De asemenea, am implementat un mecanism de limitare a numărului de cereri din partea unui singur utilizator, pentru a descuraja trimitera unui număr ridicat de cereri într-un interval scurt. În urma filtrării, acestea sunt redirecționate către <http://uv-app:8000>, unde *uv-app* este numele containerului care rulează serverul Uvicorn.

Capitolul 4

Direcții viitoare și concluzii

4.1 Direcții viitoare

A fost dificil de atins un echilibru optim între subînvățare și supraînvățare. Performanța maximă a modelului prezentat în lucrare, cu o acuratețe de 92,45% pe setul de antrenare, sugerează că există în continuare potențial pentru îmbunătățiri. O posibilă cauză a stagnării ar putea fi regularizarea excesivă, utilizând augmentare puternică a datelor, dropout, label smoothing și weight decay.

Privind modelul MobileNetV3Large, este necesară o investigație amănunțită pentru a detecta cauza scăderii semnificative a eficienței modelului în urma conversiei în format compatibil cu Android. Se pot lua în considerare utilizarea altor instrumente sau metode de conversie.

Referitor la setul de date, acesta ar putea avea imagini incorecte, întrucât imaginile au fost filtrate manual, acțiune predispusă la eroarea umană. În plus, anumite litere au exemple din multiple dialecte, ceea ce ar putea induce în eroare modelul. O metodă de a combate acest lucru este păstrarea unui singur dialect pentru fiecare literă și antrenarea folosind ponderi pentru clase, astfel încât clasele subrepräsentate să poată avea o pondere mai mare decât cele cu un număr mai ridicat de exemple.

Aplicația Android poate avea mai multe funcționalități, în special pentru utilizatorii conectați. Aceștia ar putea avea sesiunile de învățare salvate, cu posibilitatea de a le analiza și observa o evoluție a progresului. De asemenea, poate fi introdusă o secțiune cu sesiuni de antrenament specializate, cu literele la care utilizatorul prezintă probleme. Corectitudinea sesiunii de antrenament ar putea fi bazată pe siguranța modelului. Cu cât predicția este mai sigură, cu atât litera a fost mai aproape de realitate, iar bazat pe siguranța aceasta, utilizatorul ar putea primi sugestiile menționate anterior.

4.2 Concluzii

Lucrarea de licență prezentată a avut ca scop reducerea dificultăților de comunicare dintre persoanele cu deficiențe de auz și/sau vorbire și populația generală. Soluția propusă constă într-o aplicație mobilă, creată pentru sistemul de operare Android și are la bază o rețea neuronală convoluțională.

Centrul acestei aplicații este un model CNN utilizat în recunoașterea alfabetului limbajului semnelor american. Considerăm că, prin crearea unui set de date propriu, obținut prin combinarea mai multor surse, am adus o contribuție literaturii de specialitate și domeniului experimental datorită diversității imaginilor. Arhitectura propusă este una eficientă și combină elemente simple și de bază din domeniile vederii artificiale și învățării automate supervizate. Modelul atinge o acuratețe de 91,06% pe setul de testare, care conține imagini reprezentative pentru lumea reală. Pe un dispozitiv **Samsung Galaxy S9**, timpul de inferență este de aproximativ 120ms, timp care include și extragerea palmei prin MediaPipe Hands (12ms), astă că poate fi utilizat în timp real, fără întârzieri evidente majore.

În concluzie, lucrarea și-a atins obiectivul de a dezvolta o aplicație funcțională pentru traducerea alfabetului ASL, care totodată reprezintă și un punct de plecare pentru o aplicație avansată, capabilă să traducă cuvinte și propoziții.

Bibliografie

- [1] American Society for Deaf Children, *Free ASL Alphabet Chart*, <https://deafchildren.org/2019/06/free-asl-alphabet-chart/>, Accesat: 01.02.2025, 2019.
- [2] The Python Cryptographic Authority, *bcrypt: Modern password hashing for Python*, <https://github.com/pyca/bcrypt>, Accesat: 01.06.2025, 2012.
- [3] G. Bradski, „The OpenCV Library”, în *Dr. Dobb’s Journal of Software Tools* (2000).
- [4] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin și Alexandr A. Kalinin, „Albumentations: Fast and Flexible Image Augmentations”, în *Information* 11.2 (2020), ISSN: 2078-2489, DOI: [10.3390/info11020125](https://doi.org/10.3390/info11020125), URL: <https://www.mdpi.com/2078-2489/11/2/125>.
- [5] Tom Christie, *Uvicorn: The lightning-fast ASGI server*, <https://www.uvicorn.org/>, Accesat: 01.06.2025, 2018.
- [6] Android Developers, *CameraX: Jetpack support library for camera app development*, <https://developer.android.com/training/camerax>, Accesat: 01.06.2025, 2019.
- [7] Arun Kumar Dubey și Vanita Jain, „Comparative study of convolution neural network’s relu and leaky-relu activation functions”, în *Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018*, Springer, 2019, pp. 873–880.
- [8] Google, *Google Colaboratory*, 2023, URL: <https://colab.research.google.com/>.
- [9] Google Developers, *Jetpack Compose*, <https://developer.android.com/jetpack/compose>, Accesat: 01.06.2025, 2024.
- [10] The PostgreSQL Global Development Group, *PostgreSQL: The world’s most advanced open source relational database*, <https://www.postgresql.org/>, Accesat: 01.06.2025, 1996.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren și Jian Sun, „Deep residual learning for image recognition”, în *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren și Jian Sun, „Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”, în *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
- [13] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le și Hartwig Adam, *Searching for MobileNetV3*, 2019, arXiv: [1905.02244 \[cs.CV\]](https://arxiv.org/abs/1905.02244), URL: <https://arxiv.org/abs/1905.02244>.
- [14] J. D. Hunter, „Matplotlib: A 2D graphics environment”, în *Computing in Science & Engineering* 9.3 (2007), pp. 90–95, DOI: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55).
- [15] MagicStack Inc., *asyncpg: A fast PostgreSQL Database Client Library for Python/asyncio*, <https://github.com/MagicStack/asyncpg>, Accesat: 01.06.2025, 2016.
- [16] Sergey Ioffe și Christian Szegedy, „Batch normalization: Accelerating deep network training by reducing internal covariate shift”, în *International conference on machine learning*, pmlr, 2015, pp. 448–456.
- [17] Haris Iqbal, *HarisIqbal88/PlotNeuralNet v1.0.0*, versiunea v1.0.0, Dec. 2018, DOI: [10.5281/zenodo.2526396](https://doi.org/10.5281/zenodo.2526396), URL: <https://doi.org/10.5281/zenodo.2526396>.
- [18] JetBrains, *Kotlin Programming Language*, <https://kotlinlang.org>, 2011.
- [19] Michael B. Jones, John Bradley și Nat Sakimura, *JSON Web Token (JWT)*, RFC 7519, Mai 2015, DOI: [10.17487/RFC7519](https://www.rfc-editor.org/info/rfc7519), URL: <https://www.rfc-editor.org/info/rfc7519>.
- [20] Raimundo Farrapo Pinto Junior și Ialis Cavalvante de Paula Junior, *Static Hand Gesture ASL Dataset*, 2019, DOI: [10.21227/gzpc-k936](https://dx.doi.org/10.21227/gzpc-k936), URL: <https://dx.doi.org/10.21227/gzpc-k936>.
- [21] Diederik P. Kingma și Jimmy Ba, *Adam: A Method for Stochastic Optimization*, 2017, arXiv: [1412.6980 \[cs.LG\]](https://arxiv.org/abs/1412.6980), URL: <https://arxiv.org/abs/1412.6980>.
- [22] Jerome Kingsly, *American Sign Language (ASL) Alphabet Dataset*, <https://www.kaggle.com/datasets/jeromekingsly/sign-language-recognition-using-computer-vision>, Accesat: 01.02.2025, 2022.
- [23] Kapil Londhe, *American Sign Language*, <https://www.kaggle.com/datasets/kapillondhe/americansignlanguage>, Accesat: 01.02.2025, 2021.
- [24] Ilya Loshchilov și Frank Hutter, *Decoupled Weight Decay Regularization*, 2019, arXiv: [1711.05101 \[cs.LG\]](https://arxiv.org/abs/1711.05101), URL: <https://arxiv.org/abs/1711.05101>.
- [25] TorchVision maintainers și contributors, *TorchVision: PyTorch’s Computer Vision library*, <https://github.com/pytorch/vision>, 2016.

- [26] Kevin McGuinness și Sarah O’Gara, „Comparing Data Augmentation Strategies for Deep Image Classification”, în *Proceedings of the Irish Machine Vision and Image Processing Conference (IMVIP)*, Dublin, Ireland, Aug. 2019, ISBN: 978-0-9934207-4-0.
- [27] Dirk Merkel, „Docker: lightweight linux containers for consistent development and deployment”, în *Linux journal* 2014.239 (2014), p. 2.
- [28] M Mesran, Sitti Rachmawati Yahya, Fifto Nugroho, Agus Perdana Windarto et al., „Investigating the Impact of ReLU and Sigmoid Activation Functions on Animal Classification Using CNN Models”, în *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)* 8.1 (2024), pp. 111–118.
- [29] Reza Moradi, Reza Berangi și Behrouz Minaei, „A survey of regularization strategies for deep models”, în *Artificial Intelligence Review* 53.6 (2020), pp. 3947–3986.
- [30] Szymon Olewniczak, Kacper Witczak, Iwo Czartowski și Henryk Wołek, *The American Sign Language alphabet*, 2024, DOI: [10.34808/ctjj-fw17](https://doi.org/10.34808/ctjj-fw17), URL: <https://mostwiedzy.pl/en/open-research-data/the-american-sign-language-alphabet>,730112114714428-0.
- [31] Angel G. Ortiz, *American Sign Language*, <https://www.kaggle.com/datasets/angelgortiz/american-sign-language>, Accesat: 01.02.2025, 2022.
- [32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai și Soumith Chintala, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, 2019, arXiv: [1912.01703 \[cs.LG\]](https://arxiv.org/abs/1912.01703), URL: <https://arxiv.org/abs/1912.01703>.
- [33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot și E. Duchesnay, „Scikit-learn: Machine Learning in Python”, în *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [34] Nicolas Pugeault și Richard Bowden, „Spelling it out: Real-time ASL fingerspelling recognition”, în *2011 IEEE International conference on computer vision workshops (ICCV workshops)*, Ieee, 2011, pp. 1114–1119.
- [35] Python Core Team, *Python: A dynamic, open source programming language*, Python version 3.12, Python Software Foundation, 2023, URL: <https://www.python.org/>.
- [36] Sebastián Ramírez, *FastAPI: Fast and efficient web framework for building APIs with Python 3.6+*, <https://fastapi.tiangolo.com/>, Accesat: 01.06.2025, 2018.

- [37] Sebastián Ramírez, *SQLModel: SQL Databases in Python, designed for simplicity, compatibility, and robustness*, <https://sqlmodel.tiangolo.com/>, Accesat: 01.06.2025, 2021.
- [38] Miguel Rivera, *ASLYset*, <https://data.mendeley.com/datasets/xs6mvhx6rh/1>, Accesat: 01.02.2025, 2019, DOI: [10.17632/xs6mvhx6rh.1](https://doi.org/10.17632/xs6mvhx6rh.1).
- [39] Salvatore Sanfilippo și Redis contributors, *Redis: In-memory data structure store*, <https://redis.io/>, Accesat: 01.06.2025, 2009.
- [40] Jia Shijie, Wang Ping, Jia Peiyi și Hu Siping, „Research on data augmentation for image classification based on convolution neural networks”, în *2017 Chinese Automation Congress (CAC)*, 2017, pp. 4165–4170, DOI: [10.1109/CAC.2017.8243510](https://doi.org/10.1109/CAC.2017.8243510).
- [41] Karen Simonyan și Andrew Zisserman, „Very deep convolutional networks for large-scale image recognition”, în *arXiv preprint arXiv:1409.1556* (2014).
- [42] Moniciai Smith, *Asl sign language: Learning how to sign the alphabet*, <https://www.youtube.com/watch?v=8XgYji7WFuY>, Video, licențiat sub Creative Commons Attribution, 2020.
- [43] Ahmed Sowrow, Ab. Rahim, Md Iftekhar Alam Sarker Iftekhar, Sadia Islam Prova și Mohammad Rezwanul Huq, *Images of American Sign Language (ASL) Alphabet Gestures*, <https://data.mendeley.com/datasets/48dg9vhmyk/2>, Accesat: 01.02.2025, 2024, DOI: [10.17632/48dg9vhmyk.2](https://doi.org/10.17632/48dg9vhmyk.2), URL: <https://doi.org/10.17632/48dg9vhmyk.2>.
- [44] Inc. Square, *Retrofit*, <https://square.github.io/retrofit/>, Accesat: 01.06.2025, 2013.
- [45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever și Ruslan Salakhutdinov, „Dropout: a simple way to prevent neural networks from overfitting”, în *J. Mach. Learn. Res.* 15.1 (Ian. 2014), pp. 1929–1958, ISSN: 1532-4435.
- [46] Sydnee Stokes, *Alphabet in Sign Language Edited*, https://www.youtube.com/watch?v=U4M_xh-g7EM, Video, licențiat sub Creative Commons Attribution, 2018.
- [47] Igor Sysoev și Inc. Nginx, *NGINX: High-performance HTTP server and reverse proxy*, <https://nginx.org/>, Accesat: 01.06.2025, 2004.
- [48] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens și Zbigniew Wojna, „Rethinking the Inception Architecture for Computer Vision”, în *CoRR* abs/1512.00567 (2015), arXiv: [1512.00567](https://arxiv.org/abs/1512.00567), URL: [http://arxiv.org/abs/1512.00567](https://arxiv.org/abs/1512.00567).

- [49] Joshua Tauberer, *email-validator: A robust email syntax and deliverability validator*, <https://github.com/JoshData/python-email-validator>, Accesat: 01.06.2025, 2018.
- [50] SigNN Team, *ASL Sign Language Alphabet Pictures [Minus J, Z]*, <https://www.kaggle.com/datasets/signnteam/asl-sign-language-pictures-minus-j-z>, Accesat: 01.02.2025, 2020.
- [51] Jordi Viader, *American Sign Language Alphabet (Static)*, <https://www.kaggle.com/datasets/jordiviader/american-sign-language-alphabet-static>, Accesat: 01.02.2025, 2020.
- [52] Deirdre Wade, *ASL Alphabet for Dummies (i.e., hearing people)*, https://www.youtube.com/watch?v=Sejted0_MJk, Video, licențiat sub Creative Commons Attribution, 2020.
- [53] Michael L. Waskom, „seaborn: statistical data visualization”, în *Journal of Open Source Software* 6.60 (2021), p. 3021, DOI: [10.21105/joss.03021](https://doi.org/10.21105/joss.03021), URL: <https://doi.org/10.21105/joss.03021>.
- [54] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe și Youngjoon Yoo, „CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features”, în *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.
- [55] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang și Matthias Grundmann, *MediaPipe Hands: On-device Real-time Hand Tracking*, 2020, arXiv: [2006.10214 \[cs.CV\]](https://arxiv.org/abs/2006.10214), URL: <https://arxiv.org/abs/2006.10214>.
- [56] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin și David Lopez-Paz, „mixup: Beyond Empirical Risk Minimization”, în *CoRR* abs/1710.09412 (2017), arXiv: [1710.09412](https://arxiv.org/abs/1710.09412), URL: [http://arxiv.org/abs/1710.09412](https://arxiv.org/abs/1710.09412).