UNIVERSITATEA TEHNICĂ GHEORGHE ASACHI DIN IAȘI
FACULTATEA DE AUTOMATICA ȘI CALCULATOARE
PROGRAMUL DE STUDII DE MASTER: SDTW

Vedere artificială - Fire detection in natural scenes

Baraboi Gabriel - gabriel.baraboi@student.tuiasi.ro
Baraboi Ion - ion.baraboi@student.tuiasi.ro

IAȘI 2024

Cuprins

1 Introducere **3**
1.1 Scopul Lucrării 3
1.2 Implementare 3

2 Aspecte teoretice **5**
2.1 TensorFlow 5
2.2 Pandas 5
2.3 NumPy 5
2.4 Scikit-Learn (sklearn) 6
2.5 Flask 6
2.6 Arhitectura MobileNetV2 6
2.6.1 Blocuri de Reziduuri Inversate 6
2.6.2 Convoluții Separabile în Profunzime 6
2.6.3 Avantajele MobileNetV2 7

3 Implementarea aplicației **8**
3.1 Pregătirea Datelor 8
3.1.1 Antrenarea Modelului 8
3.2 Dezvoltarea Interfeței Web 8
3.3 Integrarea Modelului cu Flask 9

4 Îmbunătățiri viitoare **11**

5 Concluzii **12**

Bibliografie **12**

Capitolul 1

Introducere

1.1 Scopul Lucrării

Detectarea focului în imagini este o problemă esențială în multe domenii, incluzând siguranța publică, monitorizarea mediului și securitatea industrială. Incendiile pot avea consecințe devastatoare, provocând daune materiale semnificative, pierderi de vieți omenești și distrugerea ecosistemelor. De aceea, detectarea a focului este crucială pentru a reduce impactul acestuia și pentru a permite intervenții rapide.

În contextul creșterii rapide a tehnologiilor de învățare automată și a disponibilității resurselor computaționale puternice, soluțiile de detectare a focului au devenit mai accesibile și mai eficiente. Acest proiect își propune să dezvolte un sistem de detectare a focului în imagini utilizând tehnici avansate de învățare profundă. Proiectul folosește modelul MobileNetV2 preantrenat, care este optimizat pentru eficiență și performanță ridicată, și îl ajustează (fine-tuning) pentru a identifica prezența focului în imagini.

Scopul principal al acestui proiect este de a crea un model de învățare profundă capabil să detecteze focul în imagini cu o acuratețe ridicată. Pe lângă aceasta, proiectul își propune să dezvolte o aplicație web intuitivă și ușor de utilizat, care să permită utilizatorilor să încarce imagini și să primească imediat răspunsuri privind prezența focului. Această aplicație poate fi utilizată în diverse scenarii practice, de la monitorizarea în timp real a pădurilor și a zonelor industriale, până la implementarea în sisteme de securitate rezidențiale.

1.2 Implementare

Implementarea acestui proiect presupune utilizarea unui set divers de imagini pentru antrenarea modelului, dezvoltarea și ajustarea arhitecturii rețelei neuronale și integrarea acesteia într-o interfață web folosind Flask. Proiectul are multiple obiective specifice:

1. **Colectarea și Pregătirea Datelor:** Adunarea unui set de date adecvat, care să includă imagini cu foc și imagini fără foc, și preprocesarea acestora pentru a fi utilizate în antrenarea modelului.

2. **Dezvoltarea Modelului:** Utilizarea modelului MobileNetV2 preantrenat și ajustarea acestuia pentru a se adapta la sarcina specifică de detectare a focului. Acest proces include fine-tuning-ul ultimelor straturi ale rețelei neuronale pentru a îmbunătăți precizia pe setul nostru de date.

3. **Evaluarea Performanței:** Testarea și evaluarea modelului antrenat pentru a determina acuratețea și fiabilitatea acestuia în detectarea focului. Acest pas implică utilizarea metricilor de performanță și a tehnicilor de validare.

4. **Dezvoltarea Aplicației Web:** Crearea unei aplicații web folosind Flask, care să permită utilizatorilor să încarce imagini și să primească răspunsuri de la model. Aplicația trebuie să fie ușor de utilizat și să ofere rezultate rapide și precise.

5. **Optimizare și Îmbunătățire:** Analiza performanței sistemului în diferite condiții și efectuarea ajustărilor necesare pentru a îmbunătăți acuratețea și eficiența detectării.

Capitolul 2

Aspecte teoretice

În cadrul acestui proiect, au fost utilizate mai multe biblioteci esențiale pentru implementarea și funcționarea detectării focului în imagini. Aceste biblioteci sunt descrise mai jos:

2.1 TensorFlow

TensorFlow este o bibliotecă open-source dezvoltată de Google pentru crearea și antrenarea modelelor de învățare automată și de învățare profundă. Este utilizată pe scară largă datorită flexibilității și a capacității sale de a rula pe diferite platforme, inclusiv CPU, GPU și TPU. TensorFlow oferă un ecosistem complet pentru dezvoltarea de modele complexe, incluzând instrumente pentru preprocesarea datelor, construcția și antrenarea rețelelor neuronale, precum și pentru evaluarea și optimizarea acestora. În acest proiect, TensorFlow a fost folosit pentru a realiza fine-tuning-ul modelului MobileNetV2 și pentru a crea arhitectura modelului de detectare a focului.

2.2 Pandas

Pandas este o bibliotecă de manipulare și analiză a datelor în Python. Oferă structuri de date rapide, flexibile și expresive, concepute pentru a facilita manipularea și analiza datelor structurate și semi-structurate. Pandas permite operații eficiente de citire, scriere, filtrare, agregare și transformare a datelor. Deși nu a fost folosită direct în acest proiect specific, Pandas este adesea utilizată pentru preprocesarea datelor și pentru analiza seturilor de date în contextul învățării automate.

2.3 NumPy

NumPy este o bibliotecă fundamentală pentru calculul numeric în Python. Oferă suport pentru array-uri multidimensionale și funcții matematice de bază necesare pentru efectuarea de operații rapide asupra acestor array-uri. NumPy este esențială pentru manipularea datelor numerice și pentru efectuarea de calcule eficiente, fiind integrată cu multe alte biblioteci de învățare automată. În acest proiect, NumPy a fost utilizat pentru a manipula datele de imagine înainte de a le introduce în modelul de învățare profundă.

2.4 Scikit-Learn (sklearn)

Scikit-Learn este o bibliotecă puternică pentru învățarea automată în Python. Oferă o gamă largă de algoritmi de învățare automată, inclusiv clasificare, regresie și clustering. În plus, include instrumente pentru preprocesarea datelor, selecția modelului și evaluarea performanței. Scikit-Learn este cunoscută pentru interfața sa simplă și coerentă, fiind ușor de utilizat pentru prototiparea rapidă a modelelor de învățare automată. Deși în acest proiect specific am folosit TensorFlow pentru antrenarea modelului, Scikit-Learn este adesea utilizată pentru sarcini complementare, cum ar fi împărțirea seturilor de date și evaluarea performanței modelelor.

2.5 Flask

Flask este un micro-framework web pentru Python, cunoscut pentru simplitatea și flexibilitatea sa. Este utilizat pentru a crea aplicații web și API-uri. Flask permite dezvoltarea rapidă a aplicațiilor web printr-o arhitectură modulară care încurajează utilizarea extensiilor pentru funcționalități suplimentare. În acest proiect, Flask a fost folosit pentru a construi o interfață web simplă care permite utilizatorilor să încarce imagini și să primească răspunsuri de la modelul de detectare a focului. Aceasta a facilitat integrarea modelului de învățare profundă într-o aplicație accesibilă și ușor de utilizat.

2.6 Arhitectura MobileNetV2

MobileNetV2 este o arhitectură de rețea neuronală convoluțională dezvoltată de Google, optimizată pentru eficiență și performanță ridicată, în special pe dispozitive mobile și embedded. Arhitectura MobileNetV2 este caracterizată de utilizarea straturilor de convoluție separabile în profunzime și a blocurilor de reziduuri inversate.

2.6.1 Blocuri de Reziduuri Inversate

Blocurile de reziduuri inversate sunt componentele fundamentale ale arhitecturii MobileNetV2. Aceste blocuri utilizează o structură în care intrarea este mai întâi extinsă printr-o convoluție de 1x1, urmată de o convoluție separabilă în profunzime, și apoi redusă înapoi printr-o altă convoluție de 1x1. Acest design permite rețelei să mențină informația relevantă pe parcursul straturilor, reducând în același timp complexitatea computațională.

2.6.2 Convoluții Separabile în Profunzime

Convoluțiile separabile în profunzime descompun operația standard de convoluție în două etape: o convoluție în profunzime care procesează fiecare canal de intrare separat și o convoluție punctuală care combină rezultatele acestor convoluții într-un nou set de canale. Această abordare reduce semnificativ numărul de parametri și operațiuni necesare, menținând în același timp performanța modelului.

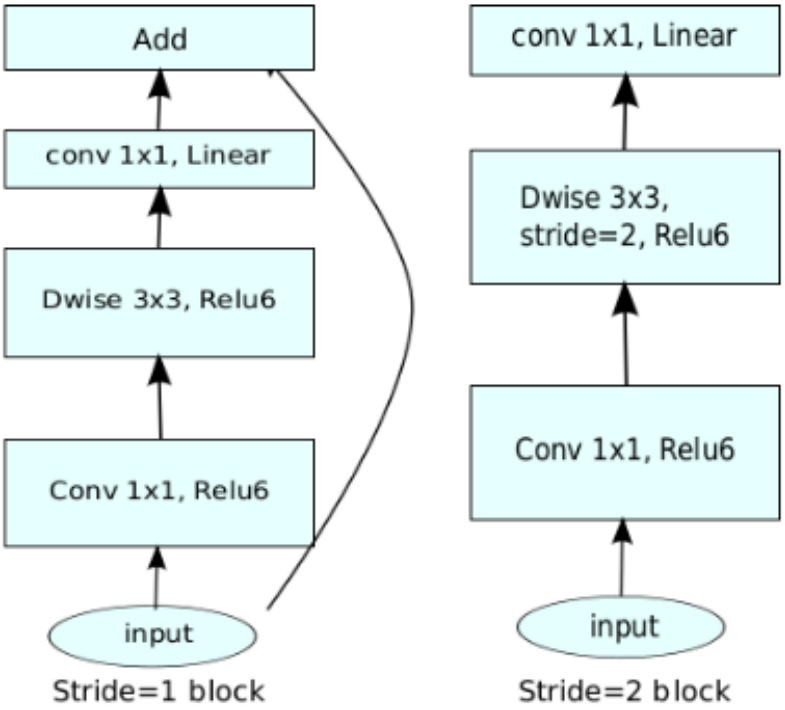

Figura 2.1: Arhitectura MobileNetV2

2.6.3 Avantajele MobileNetV2

- **Eficiență Computațională:** Utilizarea convoluțiilor separabile în profunzime și a blocurilor de reziduuri inversate reduce semnificativ numărul de operațiuni necesare, făcând modelul mai eficient pentru dispozitive cu resurse limitate.
- **Performanță Ridicată:** În ciuda eficienței sale, MobileNetV2 menține o performanță ridicată în recunoașterea obiectelor și în alte sarcini de vizualizare computerizată.
- **Flexibilitate:** Arhitectura poate fi ajustată pentru a se potrivi cu diferite constrângeri de resurse și cerințe de performanță, făcând-o ideală pentru o gamă largă de aplicații.

În acest proiect, MobileNetV2 a fost utilizată ca bază pentru detectarea focului, beneficiind de avantajele sale de eficiență și performanță pentru a crea un model rapid și precis, potrivit pentru implementări în timp real pe diverse dispozitive.

Capitolul 3

Implementarea aplicației

Implementarea aplicației pentru detectarea focului în imagini a implicat mai multe etape, inclusiv pregătirea datelor, antrenarea modelului, dezvoltarea unei interfețe web și integrarea acestora într-un sistem funcțional.

3.1 Pregătirea Datelor

Pentru antrenarea modelului, a fost necesară colectarea unui set de date cu imagini care conțin foc și imagini care nu conțin foc. Setul de date a fost împărțit în trei subseturi: antrenare, validare și testare. Imaginile au fost preprocesate pentru a fi compatibile cu arhitectura MobileNetV2, incluzând redimensionarea și normalizarea acestora.

3.1.1 Antrenarea Modelului

Modelul MobileNetV2 a fost folosit ca bază datorită eficienței și performanței sale. Codul pentru antrenarea modelului a fost adaptat din exemple disponibile pe Kaggle și a inclus următoarele etape:

- **Încărcarea MobileNetV2:** Modelul pre-antrenat MobileNetV2 a fost încărcat cu greutăți pre-antrenate pe ImageNet.
- **Fine-Tuning:** Straturile superioare ale MobileNetV2 au fost antrenate pe setul de date specific de imagini cu două etichete: foc (fire) și non-foc (not fire). Am utilizat un optimizator Adam și o funcție de pierdere de entropie încrucișată pentru antrenare.
- **Evaluarea Modelului:** Performanța modelului a fost evaluată pe setul de date de validare și testare, folosind măsurători precum acuratețea, precizia, rata de reamintire și F1 score.

3.2 Dezvoltarea Interfeței Web

Pentru a permite utilizatorilor să încarce imagini și să primească răspunsuri privind prezența focului, am dezvoltat o interfață web folosind Flask.

- **Structura Proiectului:** Aplicația web a fost structurată în mod clar, cu directoare separate pentru șabloanele HTML, fișierele CSS și codul Python.

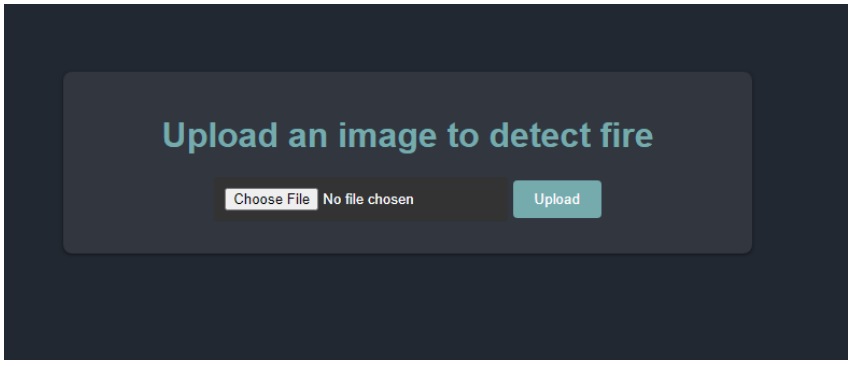

Figura 3.1: Încărcarea imaginii

- **Încărcarea Imaginilor:** Utilizatorii pot încărca imagini printr-un formular simplu. Imaginile încărcate sunt salvate temporar pe server pentru procesare.
- **Procesarea Imaginilor:** Imaginea încărcată este preprocesată și trecută prin modelul de detectare a focului. Rezultatul este afișat utilizatorului pe pagina web.

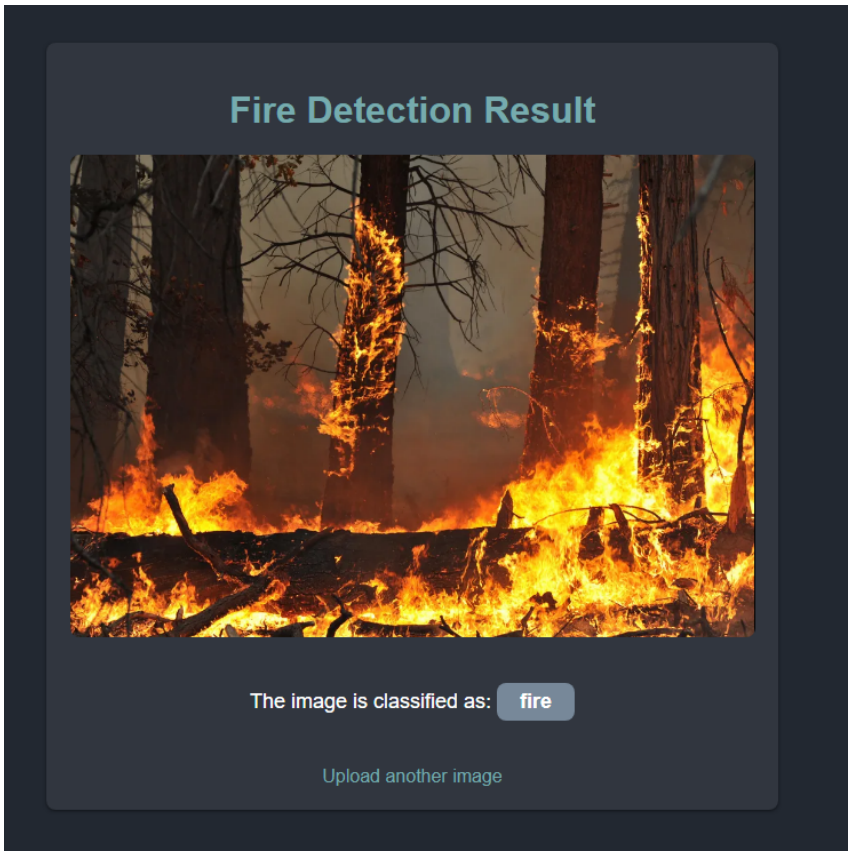

Figura 3.2: Rezultatul

3.3 Integrarea Modelului cu Flask

Integrarea modelului antrenat cu aplicația Flask a implicat următoarele etape:

- **Încărcarea Modelului:** La inițializarea aplicației Flask, modelul MobileNetV2 antrenat este încărcat în memorie.
- **Preprocesarea Imaginilor:** Imaginile încărcate sunt redimensionate și normalizate în conformitate cu cerințele modelului.
- **Clasificarea Imaginilor:** Imaginile preprocesate sunt introduse în model pentru a obține predicțiile. Răspunsul modelului este apoi trimis utilizatorului.

Capitolul 4

Îmbunătățiri viitoare

Pentru a crește performanța și utilitatea aplicației de detectare a focului, sunt planificate mai multe îmbunătățiri viitoare:

- **Extinderea Setului de Date:** Un set de date mai mare și mai diversificat ar putea îmbunătăți acuratețea modelului. Este important să includem imagini din diverse medii și condiții de iluminare pentru a face modelul mai robust.
- **Îmbunătățirea Modelului:** Explorarea altor arhitecturi de rețele neuronale sau tehnici de fine-tuning mai avansate ar putea duce la performanțe mai bune. De asemenea, antrenarea pe seturi de date mai mari și utilizarea augmentării datelor ar putea contribui la îmbunătățirea acurateței.
- **Implementarea de Funcționalități Suplimentare:** Adăugarea de funcționalități suplimentare în aplicația web, cum ar fi capacitatea de a gestiona imagini multiple simultan sau integrarea unui sistem de notificare în timp real, ar putea îmbunătăți experiența utilizatorilor.
- **Optimizarea Performanței:** Optimizarea codului și a modelului pentru a rula mai eficient pe diverse platforme, inclusiv pe dispozitive mobile, ar putea face aplicația mai accesibilă și mai practică pentru utilizarea în timp real.
- **Testare Extensivă:** Realizarea de teste extinse în diverse scenarii reale pentru a evalua și îmbunătăți robusteză modelului și a aplicației web. Feedback-ul din utilizarea în condiții reale ar putea fi valoros pentru identificarea și remedierea eventualelor probleme.
- **Securitatea Aplicației:** Asigurarea că aplicația web este securizată împotriva atacurilor cibernetice, cum ar fi injectarea de fișiere sau accesul neautorizat, este crucială pentru protejarea datelor utilizatorilor și a infrastructurii.

Implementarea acestor îmbunătățiri va contribui la dezvoltarea unui sistem de detectare a focului mai performant, mai sigur și mai util pentru utilizatorii finali, facilitând astfel intervențiile rapide și eficiente în cazurile de incendii.

Capitolul 5

Concluzii

Proiectul de detectare a focului în imagini utilizând TensorFlow și MobileNetV2 demonstrează cum tehnologiile de învățare automată pot fi utilizate pentru a rezolva probleme critice de siguranță. Prin utilizarea unui model preantrenat și aplicarea tehnicilor de fine-tuning, am reușit să dezvoltăm un detector eficient și precis pentru foc în imagini.

Integrarea modelului într-o aplicație web simplă și intuitivă, construită cu Flask, face această tehnologie accesibilă și ușor de utilizat pentru o gamă largă de aplicații practice. Aplicația permite utilizatorilor să încarce imagini și să primească rapid răspunsuri privind prezența focului, facilitând astfel intervenții rapide și eficiente.

Dezvoltările viitoare, cum ar fi extinderea setului de date, îmbunătățirea modelului și adăugarea de funcționalități suplimentare, vor contribui la creșterea performanței și utilității sistemului. Optimizarea pentru diverse platforme și asigurarea securității aplicației sunt, de asemenea, aspecte esențiale pentru a face aplicația mai robustă și mai fiabilă.

În concluzie, acest proiect oferă o soluție practică și eficientă pentru detectarea focului în imagini, demonstrând potențialul uriaș al tehnologiilor de învățare automată în îmbunătățirea siguranței și protecției împotriva incendiilor.

Bibliografie

[1] https://www.kaggle.com/

[2] https://flask.palletsprojects.com/en/3.0.x/

[3] https://www.tensorflow.org/learn