UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informaţiei
SPECIALIZAREA: Sisteme Distribuite și Tehnologii Web

Digit Recognition

Proiect Limbaje și Tehnologii Web

Scînteie Gabriel-Alexandru
Grupa 1A, SDTW

2024

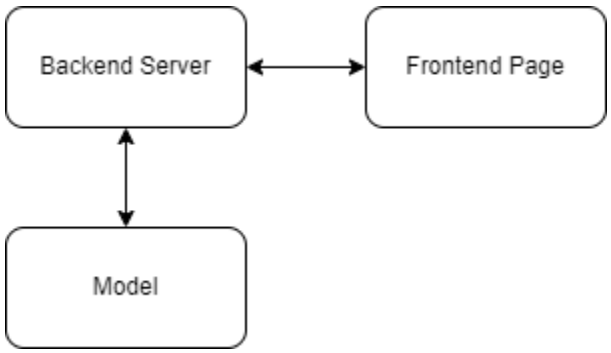
1. Introducere

1.1. Scopul proiectului

Scopul principal al proiectului este realizarea unui aplicații web care să permită clasificarea de cifre desenate de către utilizatori într-o pagină web utilizând limbajul Python și framework-urile și bibliotecile puse la dispoziție de acesta. Pentru acest lucru se utilizează o rețea neuronală convoluțională antrenată local pe setul de date MNIST, o colecție de 60000 de poze cu cifre.

1.2. Arhitectura aplicației

Aplicația este formată dintr-o pagină Web în care utilizatorul desenează cifra ce dorește să fie clasificată și unde va primi rezultatul clasificării(împreună cu o distribuție de probabilități ce indică nivelul de încredere al clasificării), un server de backend ce primește request-urile primite, le preprocesează, iar apoi le trimite unui model antrenat în prealabil să clasifice imaginile în cifre. .



Arhitectura aplicației

2. Aspecte teoretice

2.1. Python

Python este un limbaj de programare interpretat, de nivel înalt, lansat pentru prima dată în 1991. A câștigat popularitate rapidă datorită sintaxei simple, citirii ușoare și versatilității sale, permițând dezvoltatorilor să abordeze o gamă largă de probleme, de

la dezvoltarea web la inteligența artificială. În continuare voi povesti despre avantajele utilizării limbajului Python.

Python este cunoscut pentru sintaxa sa clară și concisă, care încurajează scrierea de cod ușor de înțeles. Este un limbaj potrivit atât pentru programatori începători, cât și pentru cei experimentați.

Python este un limbaj interpretat, ceea ce înseamnă că nu este necesară compilarea înainte de a rula. Acest aspect face ca dezvoltarea să fie rapidă și flexibilă. De asemenea, Python este portabil și funcționează pe orice platforme, inclusiv Windows, macOS, Linux sau chiar pe microcontrolere.

Python are o bibliotecă standard impresionantă care oferă module și funcționalități pentru o gamă largă de sarcini. Aceasta include suport pentru manipularea de fișiere, lucru cu rețele, procesare de date, criptografie, testare, inteligență artificială și multe altele. De asemenea are una dintre cele mai active comunități de dezvoltatori. Aceasta contribuie la numeroase proiecte open-source și oferă resurse, forumuri și documentație bogată. Acest suport extins facilitează învățarea și rezolvarea problemelor. Există numeroase framework-uri populare în Python, cum ar fi Django pentru dezvoltarea web, Flask pentru aplicații web mai mici, TensorFlow și PyTorch pentru machine learning, Pandas pentru manipularea datelor etc.

2.2. Rețele convoluționale

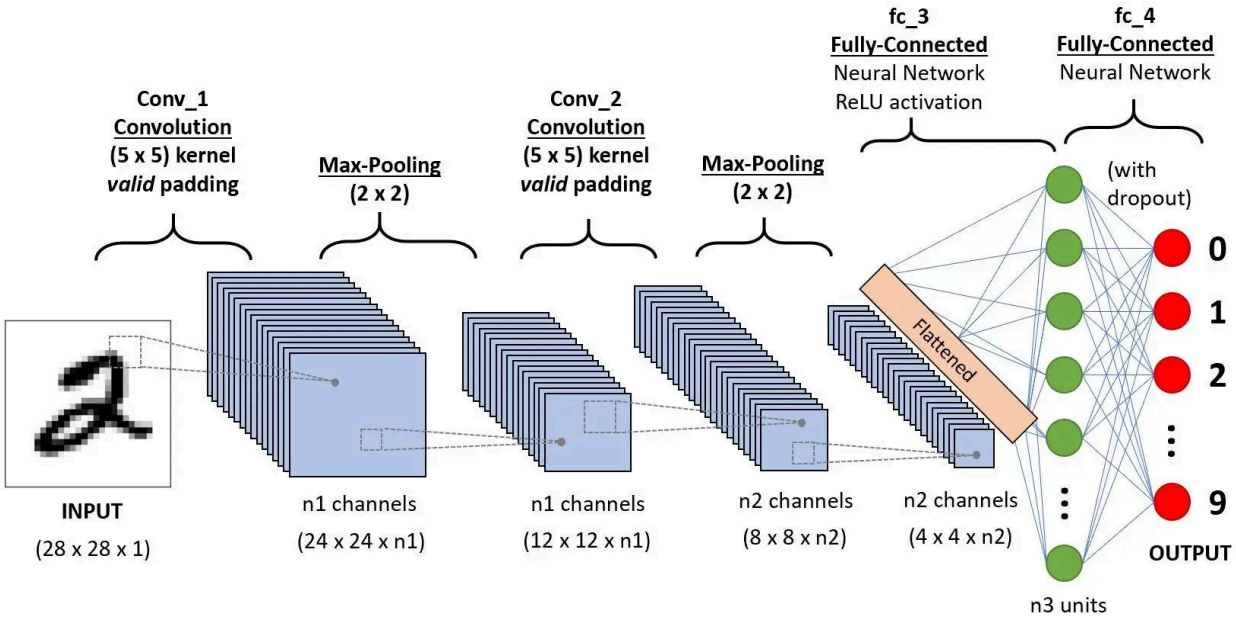
Rețelele convoluționale au devenit esențiale în domeniul recunoașterii de cifre, datorită abilității lor de a captura și înțelege caracteristici complexe din imagini. Acest subcapitol explorează aspectele teoretice ale utilizării rețelelor convoluționale în contextul recunoașterii de cifre.

Rețelele convoluționale utilizează operații de convoluție pentru a extrage caracteristici relevante din imagini. Straturile de convoluție conțin filtre care se deplasează pe întreaga imagine pentru a detecta tipare locale. În plus, straturile de pooling (cum ar fi max pooling) sunt adesea utilizate pentru a reduce dimensionalitatea și a conserva caracteristicile semnificative.

Arhitectura tipică a CNN-urilor pentru recunoașterea de cifre este următoarea:

- Straturi convoluționale: detectează caracteristici precum marginile, colțurile, sau texturile în imagini
- Straturi de pooling: realizează subesantionarea și reduce dimensiunea spațială a imaginii, păstrând trăsăturile dominante
- Straturi fully connected: pe baza caracteristicilor rezultate din straturile anterioare realizează clasificarea finală a cifrelor

Funcțiile de activare, cum ar fi ReLU (Rectified Linear Unit), sunt aplicate în mod tradițional după operații precum convoluții. Aceste funcții introduc non-linearitate în rețea, facilitând învățarea de relații complexe între caracteristici.



Arhitectura rețelei

3. Frameworks

Un framework în programare este un set predefinit de reguli, convenții și instrumente care oferă o structură și funcționalități esențiale pentru dezvoltarea rapidă și eficientă a unei aplicații software. Acesta furnizează un cadru de lucru pentru dezvoltatori, eliminând necesitatea de a scrie cod repetitiv și permițându-le să se concentreze asupra logicii esențiale ale aplicației. Framework-urile sunt proiectate pentru a facilita dezvoltarea, testarea și mentenanța aplicațiilor prin furnizarea de soluții convenționale și structuri bine definite.

Ele pot acoperi o gamă largă de domenii, inclusiv dezvoltarea web, aplicații mobile, inteligența artificială, baze de date, și multe altele. Utilizarea unui framework oferă beneficii precum coerență în codul sursă, eficiență în dezvoltare, scalabilitate și susținere a unor paradigme specifice de proiectare a software-ului. De asemenea, framework-urile sunt adesea însoțite de comunități active de dezvoltatori și documentație extinsă, ceea ce facilitează colaborarea și învățarea continuă.

Framework-urile principale folosite pentru crearea aplicației sunt PyTorch și Flask.

3.1. Pytorch

PyTorch reprezintă un framework de învățare automată, fundamentat într-o arhitectură avansată și modulară, proiectat pentru a facilita dezvoltarea, antrenarea și evaluarea modelelor de învățare automată. Dezvoltat de Facebook, PyTorch se evidențiază prin abordarea sa flexibilă și prietenoasă cu utilizatorul. PyTorch oferă un grad ridicat de control și vizibilitate în procesul de construire a modelelor, facilitând experimentele și dezvoltarea iterativă. Capacitățile sale extinse includ suportul pentru rețele neurale profunde, instrumente pentru prelucrarea datelor și facilități pentru scalarea la dimensiuni mari. Comunitatea activă și susținerea continuă din partea comunității științifice îl consolidează pe PyTorch drept un instrument esențial în peisajul învățării automate și cercetării în domeniul inteligenței artificiale.Pe lângă acestea, PyTorch oferă suport integrat pentru calcul paralel și poate beneficia de performanțe îmbunătățite prin utilizarea unităților de procesare grafică (GPU), accelerând antrenarea modelelor.

3.2. Flask

Flask este un framework web minimalist și eficient pentru limbajul de programare Python, recunoscut pentru simplitatea sa și abordarea sa modulară. Flask oferă dezvoltatorilor flexibilitate și control asupra structurii și componentelor aplicațiilor web. Oferă o metodă directă și adaptabilă pentru dezvoltarea aplicațiilor web și API-urilor bazate în Python. Flask este cunoscut pentru designul său simplu, care oferă dezvoltatorilor libertatea de a selecta elementele dorite și de a personaliza aplicațiile pentru a se potrivi nevoilor lor. Cu o arhitectură simplistă, Flask permite dezvoltatorilor să aleagă și să integreze biblioteci și extensii în funcție de nevoile specifice ale proiectului lor. Suportul integrat pentru rutare, șabloane și manipularea cererilor HTTP face dezvoltarea rapidă și intuitivă. De asemenea, Flask dispune de o comunitate activă și o documentație comprehensivă, oferind resurse ample pentru utilizatorii de toate nivelurile de experiență. Prin abordarea sa simplă și modulară, Flask rămâne o alegere populară în comunitatea dezvoltatorilor web Python.

4. Implementare

Proiectul integrează mai multe tehnologii pentru a crea o aplicație de recunoaștere a cifrelor scrise de mână. La nivelul front-end-ului, utilizarea **HTML** și **JavaScript** furnizează structura paginii web și interactivitatea necesară. HTML defineste elemente precum canvas-ul pentru desenarea cifrelor, butoanele și zonele în care vor fi afișate rezultatele și graficele.

Serverul este construit folosind **Flask**, un framework web Python simplu și eficient. Fișierul server.py definește endpoint-urile pentru gestionarea rutei principale ('/') și a solicitărilor de încărcare a imaginilor ('/upload'). Flask facilitează comunicarea între interfața web și modelul de recunoaștere a cifrelor.

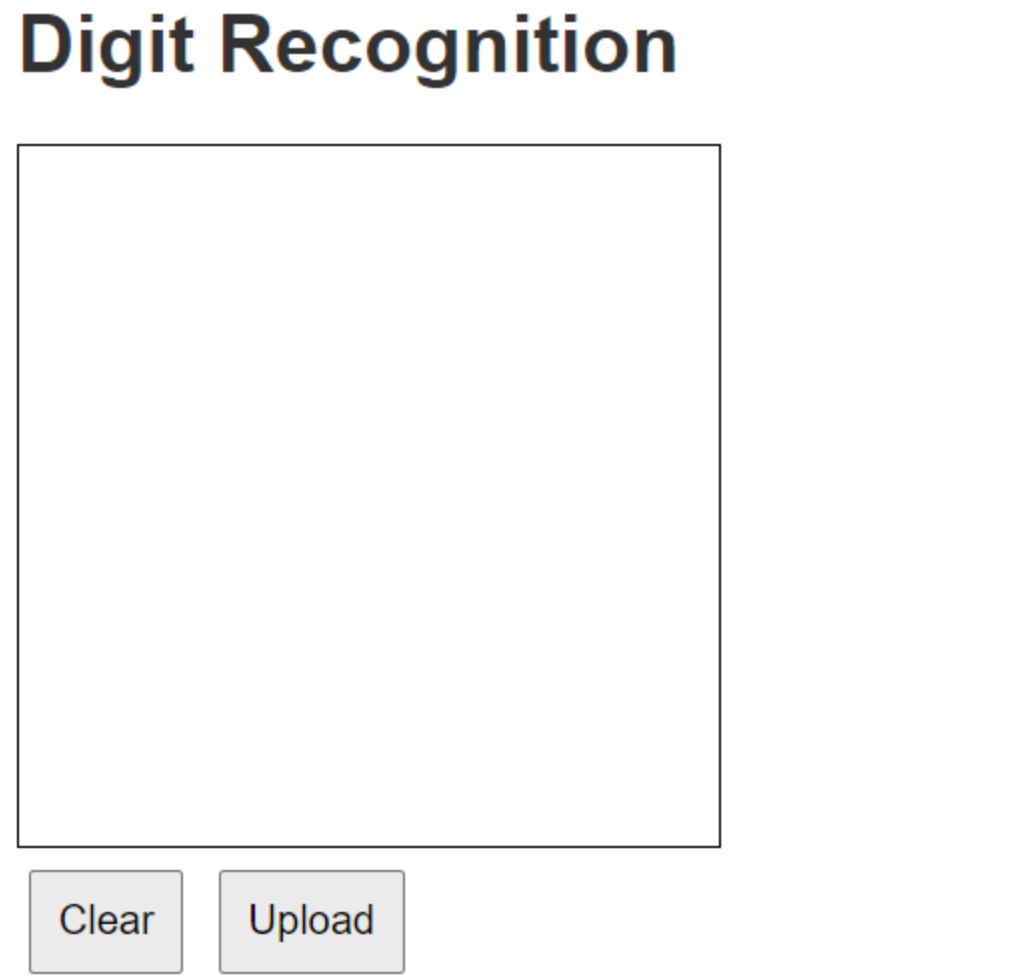
Modelul însuși, definit în model.py și încărcat la nivelului serverului prin **PyTorch**, este un model de rețea neuronală convoluțională. PyTorch furnizează funcționalitățile necesare pentru antrenarea și inferența modelului. Acesta este preîncărcat cu parametrii dintr-un fișier de stocare și este gata să realizeze predicții pe baza imaginilor primite de la utilizator. Modelul a fost în prealabil antrenat pentru a fi capabil să

Interacțiunea dintre frontend și backend este gestionată de JavaScript, care utilizează funcționalitățile canvas pentru a permite utilizatorului să deseneze cifre. Imaginile create sunt apoi convertite și trimise către server prin request-uri **HTTP**, unde sunt prelucrate și supuse inferenței utilizând modelul PyTorch.

Pe lângă acestea, proiectul utilizează biblioteca **Chart.js** pentru a afișa distribuția de probabilități sub formă de grafic bară, oferind utilizatorului o perspectivă vizuală asupra gradului de încredere al modelului în predicțiile sale.

4.1. Interfața grafică

Interfața grafică constă dintr-un canvas unde utilizatorul poate desena cifra care dorește să fie clasificată și două butoane. La apăsarea butonului de “Clear” se șterge conținutul Canvas-ului în cazul în care se dorește o nouă clasificare sau corectura unei greșeli. La apăsarea butonului de “Upload” se realizează o cerere asincronă de POST către ruta /upload.



Interfața grafică înaintea predicției

În urma primirii răspunului cererii către backend, în interfață se va afișa atât rezultatul clasificării, cât și un grafic ce reprezintă distribuția de probabilități a predicției. Înăltimea unei coloane ne spune de fapt cât de încrezător este modelul în rezultatul predicției. Imaginea desenată în canvas este trimisă către backend codifcată sub forma unui șir Base64 datorită eficienței transferului.

Interfața grafică în urma predicției

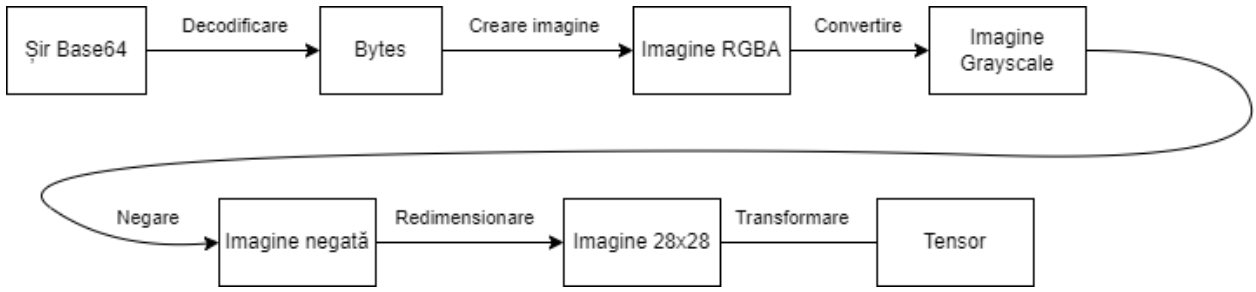
4.2. Backend

Backend-ul aplicației expune un API cu două rute:

- GET / : Rendează pagina HTML ce conține interfața grafică
- POST /upload : Primește un șir Base64 ce reprezintă imaginea desenată de utilizator și returnează un json ce conține rezultatul clasificării și distribuția de probabilități a modelului

Pentru a expune rutele descrise anterior și a manageria rendarea HTML s-a folosit framework-ul Web Flask.

Pentru a putea realiza clasificarea este necesar ca imaginea primită pe ruta /upload să fie preprocesată. Această preprocesare constă în decodificarea din Base64, recrearea imaginii din șirul de bytes utilizând biblioteca **PIL** (Python Imaging Library), convertirea imaginii din RGBA în Grayscale, negarea imaginii, redimensionarea acesteia pentru a avea mărimea 28x28 pixeli și transformarea acesteia într-un Tensor, unitatea de bază cu care funcționează Pytorch. După terminarea acestui pipeline de preprocesare, rezultatul se trece prin model pentru ca acesta să facă predicția.



Pipeline preprocesare

4.3. Modelul și antrenarea sa

Modalitatea prin care am definit structura rețelei neuronale este următoarea:

```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.fc1 = nn.Linear(64 * 7 * 7, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))

        # Facem flatten pentru a putea folosi straturile Fully Connected
        x = x.view(-1, 64 * 7 * 7)

        x = F.relu(self.fc1(x))
        x = self.fc2(x)

        x = F.log_softmax(x, dim=1)
        return x
```

Implementare definirea structurii modelului

Pentru antrenarea rețelei a fost creat un script Python separat ce downloadează setul de date MNIST și, cu ajutorul bibliotecii torch, separă datele în două DataFrame-uri, unul de antrenare și unul de testare, antrenează modelul, iar mai apoi îi verifică calitatea prin utilizarea setului de date de testare. În cazul de față, setul de date de antrenare are 60000 de cifre, iar setul de date de testare are 10000, modelul antrenat reușind să aibă o acuratețe de 99% pe setul de testare.

După antrenare parametrii modelului se salvează într-un fișier pentru a putea fi ulterior încărcate de către server fără a fi nevoie reantrenarea la fiecare rulare a serverului.

```
epochs = 10
losses = []
for e in range(epochs):
    running_loss = 0
    for images, labels in trainloader:
        optimizer.zero_grad()

        output = model(images)
        loss = criterion(output, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        average_loss = running_loss / len(trainloader)
        print(f"Training loss: {average_loss}")
        losses.append(average_loss)
```

Bucla de antrenare a modelului

```
correct_count, all_count = 0, 0
for images, labels in testloader:
    for i in range(len(labels)):
        img = images[i].view(1, 1, 28, 28)
        with torch.no_grad():
            logps = model(img)
        ps = torch.exp(logps)
        probab = list(ps.numpy()[0])
        pred_label = probab.index(max(probab))
        true_label = labels.numpy()[i]
        if (true_label == pred_label):
            correct_count += 1
        all_count += 1


print("Number of Images Tests =", all_count)
print("\nModel Accuracy =", (correct_count / all_count))

torch.save(model.state_dict(), 'trained_model.pth')
```

Bucla de testare a modelului

5. Concluzie

În concluzie, aplicația web prezentată propune o soluție la rezolvarea problemei recunoașterii de cifre desenate de mână cu ajutorul rețelelor neuronale convoluționale și pune la dispoziție o interfață grafică intuitivă ce facilitează testarea cu ușurință a modelului. Toate aceste lucruri au fost posibile cu ajutorul utilizării limbajului Python și a multiplelor biblioteci și framework-uri puse la dispoziție de acesta.

6. Bibliografie

1. https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/
2. https://flask.palletsprojects.com/en/3.0.x/
3. https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
4. https://auth0.com/blog/developing-restful-apis-with-python-and-flask/
5. https://refine.dev/blog/how-to-base64-upload/
6. https://www.w3schools.com/js/js_graphics_chartjs.asp
7. https://www.w3schools.com/graphics/canvas_drawing.asp