Tema: Prepoznavanje i Klasifikacija Voća i Povrća

Autor: Tihomir Stojković, Minja Stankov

Datum: Septembar 2024

Univerzitet u Beogradu

Matematički fakultet

Sadržaj

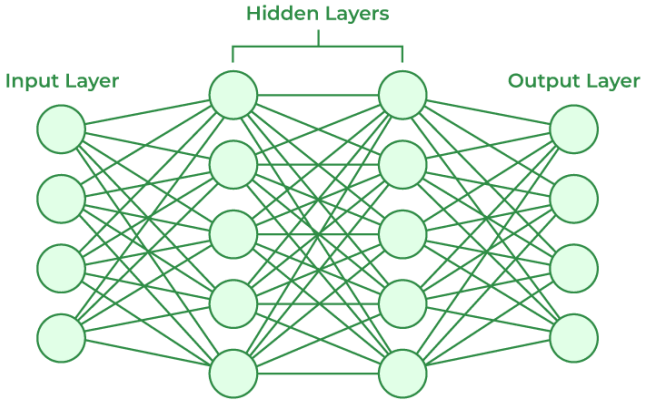
1 Uvod

U poslednjih nekoliko godina, napredak u računarskoj inteligenciji i mašinskom učenju unapredio je različite sektore, uključujući poljoprivredu i preradu hrane. Jedna intrigantna primena je razvoj sistema zasnovanih na neuronskim mrežama koji su sposobni da prepoznaju i kategorizuju voće i povrće. Takvi sistemi pružaju značajan potencijal za poboljšanje efikasnosti, kontrole kvaliteta i konkurentnosti na tržištu.

Ovaj rad predstavlja Python projekat koji ima za cilj da iskoristi neuronske mreže kako bi prepoznao i kategorizovao različite vrste voća i povrća. Korišćenjem algoritama dubokog učenja, naš projekat ima za cilj da automatizuje proces klasifikacije, čime se olakšavaju zadaci koji su do sada zavisili od ljudske procene.

2 Neuronske mreže

Neuronske mreže predstavljaju jedan od najstarijih i najuspešnijih pristupa mašinskom učenju. Datiraju iz 50-ih godina dvadesetog veka. Njihov tekući uspon počinje 2006. godine sa pojavom takozvanih dubokih neuronskih mreža, a novi zalet dobijaju 2012. godine kada se duboke mreže koriste u rešavanju problema računarskog vida, čiji je cilj da oponaša sposobnost ljudskog vizuelnog sistema da percipira, analizira i interpretera vizuelne informacije iz okoline. Od tada su neuronske mreže postigle velike uspehe i u mnogim drugim oblastima, primarno u obradi prirodnog jezika, obradi govora, igranju igara, itd. U nekim problemima, prevazišle su i mogućnosti ljudskih eksperata.



Postoje različite vrste neuronskih mreža, često specijalizovanih za konkretne primene. Svaka mreža sastoji se od određenog broja elementarnih jedinica koje nazivamo neuronima. Neuroni, po uzoru na biološke neurone u mozgu, jedni drugima prosleđuju signale i izračunavaju nove signale na osnovu onih koji su im prosleđeni. Tačna struktura povezanosti neurona i način na koji oni vrše izračunavanja čine takozvanu arhitekturu mreže i precizno određuju o kakvoj mreži se radi. Arhitekturu mreže definiše ekspert zavisno od specifičnosti problema koji se rešava.

Neuroni su osnovne jedinice izračunavanja čijim se povezivanjem grade neuronske mreže. Oni kao ulaze primaju signale od drugih neurona i transformišu ih na neki način čime proizvode svoj izlazni signal. Svi signali predstavljeni su realnim brojevima, a njihova obrada vrši se nekom jednostavnom matematičkom funkcijom koja obično uključuje linearnu kombinaciju ulaznih signala, a potom nelinearnu transformaciju te linearne kombinacije.

3 Opis problema

Naš rad se fokusira na razvoj sistema zasnovanog na neuronskim mrežama koji je sposoban da automatizuje proces klasifikacije voća i povrća. Konkretno, cilj nam je da obučimo model dubokog učenja koristeći skup podataka koji sadrži slike različitih vrsta voća i povrća, zajedno sa odgovarajućim oznakama. Obučeni model će naučiti da razlikuje različite vrste proizvoda na osnovu vizuelnih karakteristika koje izvlači iz ulaznih slika.

4 Implementacioni pristup

Da bismo rešili zadatak klasifikacije prepoznavanja i kategorizacije voća i povrća, koristili smo kombinaciju najmodernijih arhitektura neuronskih mreža:

- Konvolucione neuronske mreže (CNN)
- Mreža sa preostalim vezama (ResNet)
- Vizualni transformatori (ViT)

Svaka arhitektura ima specifične prednosti, što omogućava primenu različitih pristupa u ekstrakciji karakteristika i učenju reprezentacija. Konvolutivne neuronske mreže (CNN) su najefikasnije za analizu slika zahvaljujući svojoj sposobnosti da prepoznaju lokalne obrasce i hijerarhije karakteristika. ResNet arhitektura se posebno ističe u rešavanju problema sa degradacijom

performansi kod dubokih mreža, omogućavajući dublje mreže bez gubitka preciznosti. Vision Transformer (ViT) mreže, s druge strane, pružaju naprednu obradu slike koristeći mehanizam pažnje, što ih čini pogodnim za zadatke koji zahtevaju globalno razumevanje slike.

5 Konvolutivne neuronske mreže

Konvolutivne neuronske mreže predstavljaju specijalizovanu vrstu neuronskih mreža dizajniranih za obradu podataka u obliku mreže, posebno slika, inspirisane vizuelnim korteksom ljudskog mozga, koristeći hijerarhijske slojeve izvlačenja karakteristika.

Ključne komponente CNN-ova:

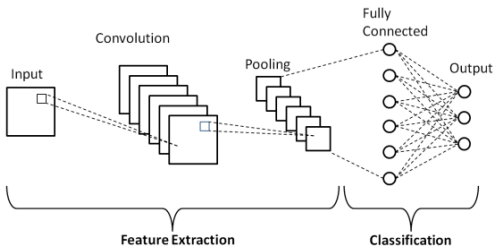
1. **Konvolutivni slojevi:** Objašnjenje kako konvolutivni slojevi primenjuju filtere za izvlačenje karakteristika iz ulaznih slika, čuvajući prostorne odnose.

2. **Slojevi agregacije:** Smanjuju veličinu karakteristika radi smanjenja složenosti obrade i ekstrahovanja dominantnih karakteristika. Obično operišu na malim regionima ulaznih karakterističnih mapa (npr. 2x2 ili 3x3) i primenjuju operaciju agregacije, poput maksimalne agregacije ili prosečne agregacije, na svaki region.

3. **Aktivacione funkcije:** Diskusija o aktivacionim funkcijama poput ReLU (Rectified Linear Unit), koje uvode nelinearnost u mrežu i pomažu u učenju karakteristika.

4. **Potpuno povezani slojevi:** Ovi slojevi povezuju neurone jednog sloja u sve neurone sledećeg sloja. Najčešće se stavljaju na kraju mreže i služe da vrše klasifikaciju na osnovu ekstrahovanih karakteristika koje smo naučili iz konvolutivnih i slojeva agregacije.

Proces treninga i optimizacije je ključan za efikasno učenje CNN-a i postizanje visokih performansi.



6 Trening i optimizacija

1. **Inicijalizacija parametara:** Pre nego što počne proces treninga, parametri CNN-a, kao što su težine i pristrasnost, se inicijalizuju nasumično ili korišćenjem neke od specifičnih tehnika inicijalizacije parametara.

2. **Propagacija unapred:** Tokom faze propagacije unapred, ulazni podaci se prosleđuju kroz neuronsku mrežu. Za svaki sloj, izračunavaju se aktivacije neurona primenom odgovarajućih funkcija na ponderisane sume ulaza. Na kraju ovog procesa generiše se izlazni rezultat.

```
[ ]  class ConvNet(nn.Module):
         def __init__(self):
             super().__init__()
             self.layer1 = nn.Conv2d(3, 32, 3)
             self.layer2 = nn.Conv2d(32, 64, 3)
             self.layer3 = nn.Conv2d(64,128,3)
             self.pool = nn.MaxPool2d(2)
             self.flatten = nn.Flatten(start_dim=1)
             self.fc1 = nn.Linear(107648,36)

         def forward(self, x):
             x = F.relu(self.layer1(x))
             x = self.pool(x)
             x = F.relu(self.layer2(x))
             x = self.pool(x)
             x = F.relu(self.layer3(x))
             x = self.pool(x)
             x = self.flatten(x)
             x = self.fc1(x)
             return x
```

```
from torch import optim
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters())
device = 'cuda' if torch.cuda.is_available() else 'cpu'
model.to(device)
num_epochs = 5
for epoch in range(num_epochs):
    train_loop(train_loader, model, loss_fn, optimizer, device)
    test_loop(test_loader, model, loss_fn, device)
```

```
  0%|          | 0/98 [00:00<?, ?batch/s]/usr/local/lib/python3.10/dist-packages/PIL/Ir
  warnings.warn(
100%|██████████| 98/98 [12:17<00:00,  7.52s/batch, accuracy=9.53, loss=0.105]
Average loss: 0.08528143250510553
Accuracy: 0.3398328690807799
100%|██████████| 98/98 [11:42<00:00,  7.17s/batch, accuracy=26.9, loss=0.0817]
Average loss: 0.05998147530143971
Accuracy: 0.520891364902507
100%|██████████| 98/98 [11:48<00:00,  7.23s/batch, accuracy=41.2, loss=0.0645]
Average loss: 0.04261568861087384
Accuracy: 0.6824512534818942
100%|██████████| 98/98 [11:55<00:00,  7.30s/batch, accuracy=52.1, loss=0.0525]
Average loss: 0.029994216885075264
Accuracy: 0.7743732590529248
100%|██████████| 98/98 [12:10<00:00,  7.45s/batch, accuracy=61.2, loss=0.0411]
Average loss: 0.027351595565137093
Accuracy: 0.8189415041782729
```

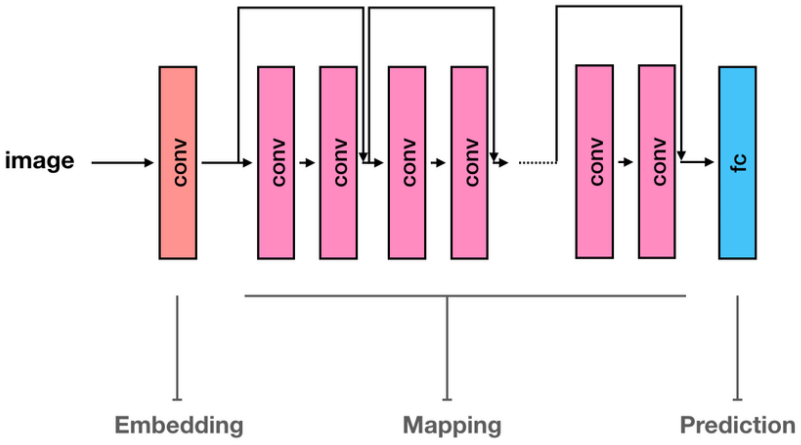
3. **Izračunavanje greške:** Nakon što se generiše izlaz, koristi se odgovarajuća funkcija greške kako bi se izračunala razlika između predviđenih rezultata i stvarnih oznaka. Ova greška služi kao metrika performansi neuronske mreže i koristi se za prilagođavanje parametara tokom procesa obuke.

4. **Propagacija unazad:** U ovoj fazi, greška se propagira unazad kroz mrežu kako bi se izračunali gradijenti funkcije greške u odnosu na parametre mreže. Koristi se pravilo lanca kako bi se efikasno propagirala greška kroz svaki sloj mreže. Ovi gradijenti se zatim koriste za prilagođavanje parametara tokom optimizacionog procesa.

5. **Optimizacija parametara:** Nakon što su izračunati gradijenti, koristi se odgovarajući optimizacioni algoritam (kao što je SGD) kako bi se prilagodili parametri mreže u pravcu koji minimizuje funkciju greške. Ovaj proces se ponavlja iterativno za više iteracija treninga.

7 Residualna Mreža

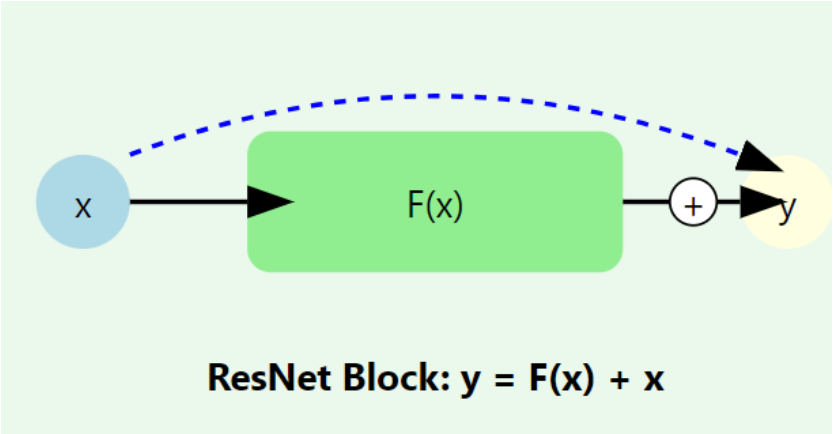
ResNet je tip arhitekture dubokih neuronskih mreža koji je uveden kako bi se rešio problem nestajućeg gradijenta u veoma dubokim konvolutivnim neuronskim mrežama (CNN-ovima). Problem nestajućeg gradijenta se javlja prilikom obučavanja dubokih mreža, posebno onih sa mnogo slojeva, i ometa sposobnost mreže da efikasno uči. Inovacija ResNet-a leži u korišćenju residualnih veza, koje omogućavaju obučavanje veoma dubokih mreža bez gubitka performansi.



Ključni koncepti u okviru ResNet-a:

1. **Rezidualni blokovi:** Osnovni gradivni blokovi. Rezidualni blok se sastoji od dva glavna puta: škraćeni"ili "identitetni"put i "glavni"put. Glavni put obično sadrži dva ili više konvolutivnih slojeva, svaki praćen slojem za normalizaciju grupa i ReLU aktivacionom funkcijom. Identitetni put je direktna veza (skok veza) od ulaza do izlaza bloka.

2. **Skok veze:** Skok veze omogućavaju mreži da uči residualne funkcije umesto direktnog učenja željenog podložnog preslikavanja. Umesto da pokušava da aproksimira željeno preslikavanje $H(x)$, gde je x ulaz u određeni sloj, ResNet uči rezidualno preslikavanje $F(x) = H(x) - x$. Izlaz rezidualnog bloka se zatim računa kao zbir ulaza i residualnog preslikavanja: izlaz = ulaz + $F(x)$.



3. **Rezidualno učenje:** Upotreba rezidualnih veza omogućava lakše obučavanje veoma dubokih mreža. Omogućavanjem gradijenta da teče direktno kroz skok veze, ResNet rešava problem nestajućeg gradijenta. To znači da čak i u veoma dubokim mrežama, gradijent može efikasno da se propagira nazad kroz slojeve tokom obučavanja, olakšavajući proces optimizacije.

4. **Arhitektura mreže:** ResNet arhitekture obično se sastoje od niza rezidualnih blokova, pri čemu svaki blok sadrži više konvolutivnih slojeva. Dubina ResNet modela može varirati u zavisnosti od konkretne primene i raspoloživih računarskih resursa. Česte varijante uključuju ResNet-18, ResNet-34, ResNet-50, ResNet-101 i ResNet-152, pri čemu brojevi označavaju ukupan broj slojeva u svakoj varijanti.

ResNet arhitekture su široko korišćene i efikasne u različitim zadacima računarske vizije, uključujući klasifikaciju slika, detekciju objekata, semantičku segmentaciju i još mnogo toga.

8 Vizuelni transformer

Odnosi se na specifičnu vrstu arhitekture neuronskih mreža koja proširuje model transformera kako bi se rukovao vizuelnim podacima. Vizuelni transformer primenjuje arhitekturu transformera direktno na prostornu rešetku piksela slike, tretirajući sliku kao sekvencu tokena. Ovo omogućava modelu da uhvati interakcije između piksela širom cele slike, bez oslanjanja na ručno izrađene osobine ili prostorne hijerarhije.

Ključni koncepti u okviru Vizuelnog transformera:

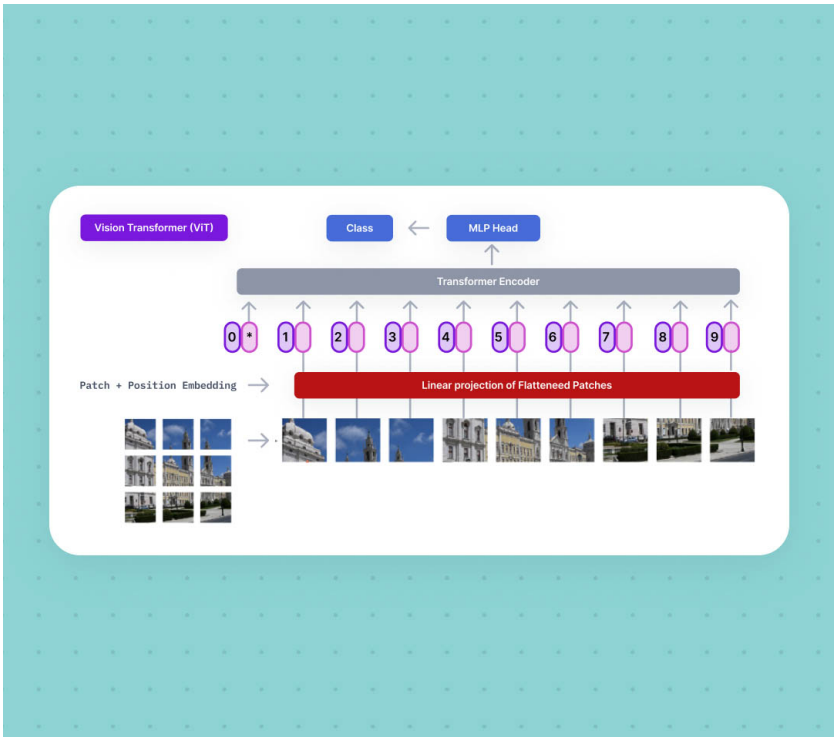
1. **Tokenizacija:** Ulazna slika se deli u rešetku delova, pri čemu se svaki deo tretira kao token. Zatim se ovi delovi linearno projektuju u visokodimenzionalne ugnežđenja.

2. **Poziciono kodiranje:** Budući da arhitektura transformera ne razume prostorne odnose između tokena, poziciona kodiranja se dodaju u ugnežđenja tokena kako bi se pružile informacije o prostornim pozicijama delova unutar slike.

3. **Transformer enkoder:** Ugnežđenja tokena, zajedno sa pozicionim kodiranjima, prolaze kroz više slojeva blokova transformatora. Svaki blok enkodera sastoji se od mehanizama pažnje više glava za samoučenje, praćenih neuronskim mrežama za feedforward. Samoučenje omogućava modelu da uhvati interakcije između različitih delova, dok feedforward mreže uvode nelinearnost.

4. **Glava za klasifikaciju:** Na izlazu skupa blokova enkodera transformera obično se dodaje glava za klasifikaciju kako bi se predvideli labeli ili obavili drugi zadaci na osnovu naučenih reprezentacija.

Obuka vizualnog transformera podrazumeva optimizaciju parametara modela (uključujući ugnežđenja tokena, poziciona kodiranja i blokove transformatora) korišćenjem propagacije unazad i gradijentnog spusta. Funkcija gubitka koja se koristi tokom obuke zavisi od specifičnog zadatka koji se obavlja. Vizuelni transformeri su pokazali impresivne performanse u različitim zadacima računarske vizije, ali takođe dolaze sa izazovima u pogledu računske složenosti, posebno kada se bave slikama visoke rezolucije, zbog kvadratne složenosti mehanizama samoučenja. Istraživači aktivno istražuju načine za



poboljšanje efikasnosti i skalabilnosti vizualnih transformera uz očuvanje njihovih performansi na vizuelnim zadacima.

9 Zaključak

U zaključku, naš rad pokazuje efikasnost korišćenja savremenih arhitektura neuronskih mreža, uključujući konvolutivne neuronske mreže (CNN), mreže sa preostalim vezama (ResNet) i vizualne transformere, za zadatak prepoznavanja i kategorizacije voća i povrća. Kroz rigorozno eksperimentisanje i analizu, pokazali smo da svaka arhitektura donosi različite prednosti procesu klasifikacije, omogućavajući sveobuhvatnu eksploraciju strategija ekstrakcije karakteristika i učenja reprezentacija.

Iskorišćavanjem jedinstvenih prednosti CNN-a, ResNet-a i vizualnih transformera, razvili smo robusni algoritamski okvir sposoban da precizno klasifikuje različite tipove proizvoda na osnovu vizuelnih karakteristika ekstrahovanih iz ulaznih slika. Naš pristup ne samo da postiže visoku tačnost klasifikacije, već takođe pokazuje svestranost i skalabilnost, čime je dobro prilagođen za primenu u stvarnim uslovima poljoprivrede.