Projekat 1 Duboko učenje

Nemanja Vujić

November 2024

1 Uvod

Primena konvolucionih mreža na problem preopoznavanja boja na slikama. Ovaj problem se bavi prepoznavanjem najdominantnije boje na slici. Za ovaj projekat koristimo podskup *House Plant Species*[1]. Podaci nisu direktno namenjeni za ovaj problem što znači da je potrebna obrada, označivanje i analiza podataka. Nakon čega je moguće preći na rešenje ovog projekta.
Primer podataka:

Pirmer podatka *(Slika)*

Dominantna boja *(rgb 142,233,254)*

[1] https://www.kaggle.com/datasets/kacpergregorowicz/house-plant-species

2 Opis podataka

Skup podataka koji se koristi za ovaj projekat nije adekvatno označen za primenu konvolucione mreže. Pre upotrebe podaci su uz pomoć algoritma obrađeni kako bi se za svaku sliku pronašla njena najdominantnija boja. Takvi podaci su nakon označivanja sačuvani u *.csv* fajlu za dalju upotrebu.

Označavanje podataka. Prvo su proverene dimenzije podataka, odnosno koliko različitih dimenzija imaju slike. Nakon čega su slike verifikovane uz pomoć *Pillow*[2] biblioteke u pythonu. Za detaljno označivanje svih fajlova korišćena je *pyCuda*[3] biblioteka kako bi paralelno izračunali dominantnu boju na većem broju podataka koristeći sledeći algoritam:

```
__global__ void count_rgb(unsigned char *image, int *counters, int
    width, int height) {
    int idx = threadIdx.x + blockIdx.x * blockDim.x;
    int idy = threadIdx.y + blockIdx.y * blockDim.y;

    if (idx < width && idy < height) {
        int pixel_index = (idy * width + idx) * 3;
        int r = image[pixel_index];
        int g = image[pixel_index + 1];
        int b = image[pixel_index + 2];

        int color_key = (r << 16) | (g << 8) | b;
        atomicAdd(&counters[color_key], 1);
    }
}
```

Kao rezultat dobijamo *.csv* tabelu. Isečak iz tabele:

Image Name	Dominant Color	Hex Color
1.jpg	(np.int64(93), np.int64(70), np.int64(78))	#5D464E
10.jpg	(np.int64(183), np.int64(177), np.int64(177))	#B7B1B1
100.jpg	(np.int64(29), np.int64(29), np.int64(37))	#1D1D25
102.jpg	(np.int64(229), np.int64(228), np.int64(226))	#E5E4E2

Prva kolona predstavlja ime slike (svaka slika u podacima ima ime kao broj slike), druga kolona je rgb vrednost najdominantnije boje slike u numpy formatu i treća kolona je Hex kod te najdominantnije boje. Ukupno ima oko 14.000 slika za model da radi na njima.

[2] https://pillow.readthedocs.io/en/stable/
[3] https://documen.tician.de/pycuda/

3 Analiza podataka

Kako bi smo znali šta tačno možemo da očekujemo za ovaj set podataka prvo ćemo izvršiti analizu celog seta kako bi dobili potrebne podatke. Za podatke uradjeno je 6 relevantnih analiza podataka:

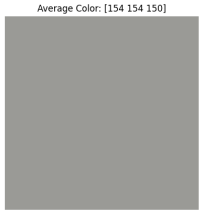
1. Detekcija previše tamnih slika
2. Prosečna dominantna boja
3. Tamne i svetle boje po r,g,b dominantnim vrednostima
4. Proporcija kanala
5. RGB klasterovanje
6. Deset unikatnih boja
7. Znatno dominantan kanal

3.1 Detekcija previše tamnih slika

Izvršena je detekcija slika kojima je dominantna RGB vrednost ispod (10, 10, 10) što znači da su ovo dosta tamne slike. Ovi tako reći izuzetci mogu negativno da utiču na sposobnost modela. Možda će biti potrebno da se slike uklone. **Rezultat ove analize je 19 unikatnih slika.**

3.2 Prosečna dominantna boja

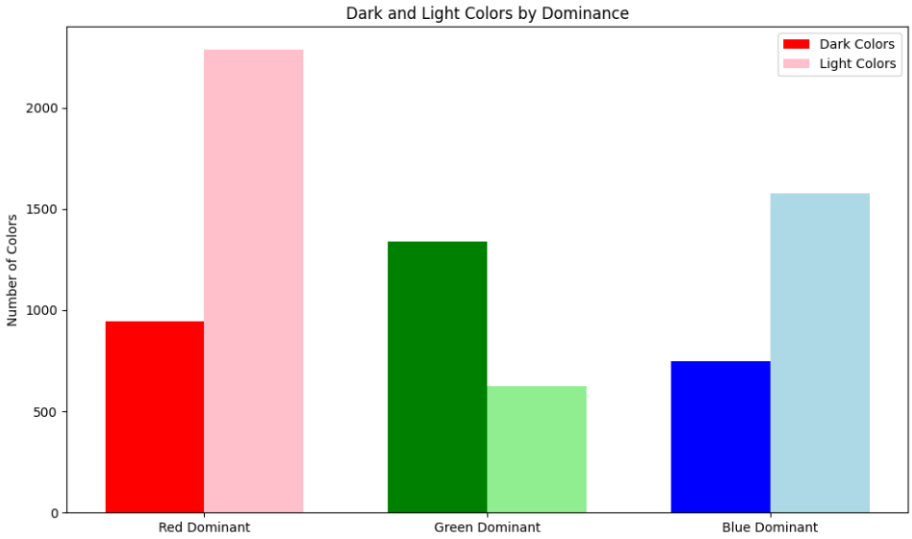
Prosečna boja kada se izvuče prosek svih dominantnih boja je blizu prirodnoj sivoj. To nam govori da je ovaj set podataka dobro podeljen na različite boje spektruma.



Prosečna boja

3.3 Tamne i svetle boje po r,g,b dominantnim vrednostima

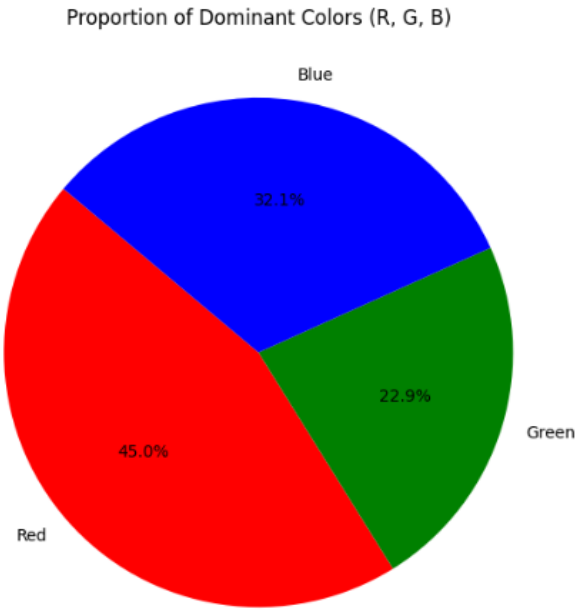
Distribucija tamnih i svetlih boja po kanalima pokazuje zadovoljavajuću variaciju. Kod slika gde su dominantni crveni i plavi kanali slike su značajno svetlije, dok u zeleno dominantnim slikama ima dosta više tamnijih slika što može da ukaže na problem kasnije. Medjutim ukupna podela tamnih i svetlih slika je za sada zadovoljavajuća.



Tamne i svetle boje po kanalima

3.4 Proporcija kanala

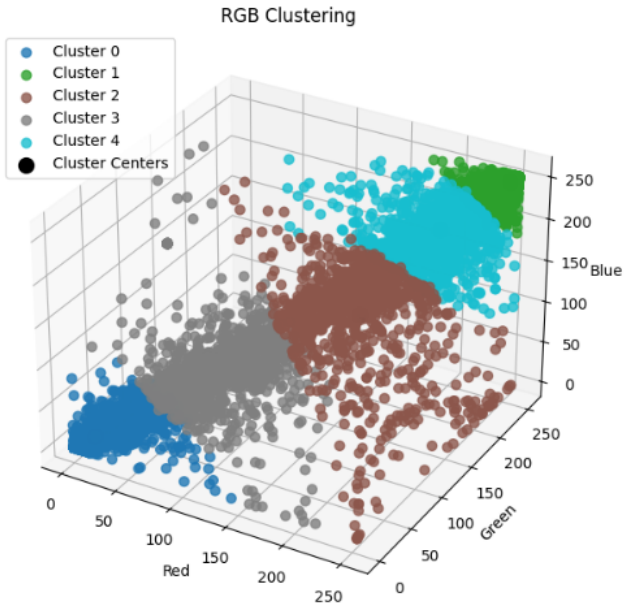
Sa ovom analizom proveravamo da li je set podataka balansiran odnosno da jedan od kanala nije značajno zastupljeniji od ostalih. Rezultati ukazuju da ima malo više slika kod kojih je crveni kanal dominantan.



Proporcija kanala

3.5 RGB klasterovanje

Klasterovanje vrednosti ukazuje na adekvatno i distinktno grupisanje u setu podataka. Klasteri mogu da pomognu modelu da nauči specivične palete boja koji se nalaze u tim klasterima.



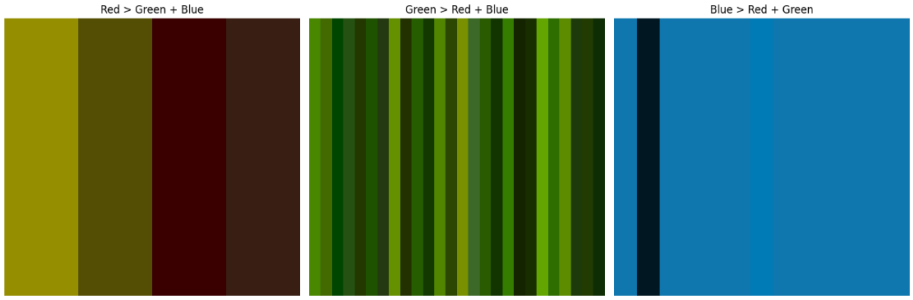
Klasteri

3.6 Deset unikatnih boja

Proverne su unikatne boje unutar seta podataka i njihove unikatne vizuelne elemente. Ove unikatne boje mogu da budu ključni feature-i u modelu. Rezultat je 10 izdvojenih slika kod kojih su dominantne boje jedinstvene.

3.7 Znatno dominantan kanal

Slike u kojima crveni, zeleni i plavi kanali dominiraju nad zbirom druga dva kanala su identifikovane, što nam pokazuje pod set slika sa jakom jednobojnom dominacijom. Ovo može u treniranju modela da značajno pomogne, a može i da škodi modelu u zavisnosti od načina uotrebe.



Dominantni kanali

4 Zaključci analize podataka

4.1 Proporcija dominantnih kanal u odnosu na slike

Proporcija dominantnih kanal u odnosu na slike je adekvatna što nam govori činjenica da su ti dominantni kanali približno podeljeni na trećine. Naime plavi kanal je 36.2%, zeleni 35.0% i crveni 28.8%. Takodje su i klasteri približnih veličina (osim jednog).

4.2 Dosta artifakta na slikama

Dosta slika (19) sadrži artifakte što dovodi do znatno tamnije dominantne boje nego na ostalim slikama. Ovaj problem se može rešiti na 2 načina: 1. Eliminisanjem tih slika iz seta podataka i 2. Sredjivanjem artifakta.

4.3 Prosečna dominantna boja

Rezultat ove analize uliva poverenje u izabrani set podataka zato što je rezultat analize sasvim očekivan. Naime očekivano je da prosečna boja bude siva sa vrednostima 127, 127, 127. Naš rezultat se poklapa pa čak i nadovezuje na ovo time što je zeleni kanal veći (očekivano kod slika vegetacije zbog dominantne zelene boje).

4.4 Tamne zelene boje

Kod zlika kod kojih je zelena dominantna boja vidimo značajan skok u tamnim slikama što može u daljem radu sa modelom da dovede to problema. Posmatraćemo ovaj problem i imati na umu kako bi ispravili ovu grešku ako dodje do problema kasnije.

4.5 Finalni podaci nakon analize

Set podataka smo nakon analize i zaključaka modifikovali da bude lakši i tačniji za rad sa modelom. Preprocesovali smo podatke tako što smo izbacili slike sa artifaktima koji znatno utiču na analizu dominantne boje (19 slika) i promenili formate u csvu iz *np.int64* u klasične brojeve, takodje smo izbacili Hex kod boje jer je bio potreban samo sa proveru. Konačni skup podataka izgleda:

Image Name	Dominant Color
1.jpg	93, 70, 78
10.jpg	183, 177, 177
100.jpg	29, 29, 37
102.jpg	229, 228, 226

5 Prva verzija LeNet modela

U početku je kao adekvatan model delovao jednostavnu leNet arhitekturu. Dva konvoluciona sloja u kojima prvi sadrži 6 filtera dimenzije 5x5 dok drugi ima 16 filtera. ReLU funkcije se koriste za uvodjenje nelinearnosti. Pooling slojevi za prosečno uzorkovanje smanjuju kompleksnost modela. I dva potpuno pobezana sloja sa 120 i 84 neurona respektivno, završavaju sa 3 izlazna neurona koji predstavljaju RGB vrednosti. Dropout slojevi dodati kako bi smanjili mogućnost overfittinga.
LeNet model je treniran na skupu podataka podeljene na 60% za trening 20% za validaciju i 20% za testiranje. Broj epoha je 20. Veličina batcha je 64 i funkcija greške mean squared error. Model je nakon svega imao tačnost od 14.01% zbog cega prelazimo na ResNet arhitekturu.

6 Prva verzija ResNet modela

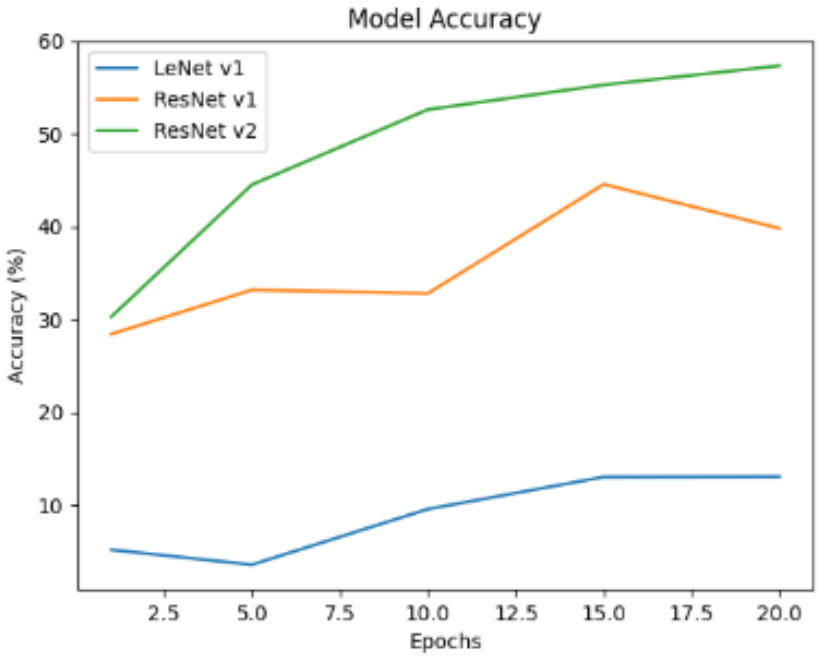
Nakon loših rezultata leNet modela odlučio sam da isprobam resNet arhitekturu. Ovaj model je kompleksniji od prethodnog ali bi trebalo da adekvatno radi na zadatom problemu. Za ResNet koriscen je built in model sa nasim parametrima. Hiperparametri koji su korišćeni su 20 epoha, veličina batcha 64, adam optimizator sa početnom stopom 0.001 i težinskom regularizacijom i funkcija greške ista kao u prethodnom modelu mean squared error. Tacnost modela na kraju testiranja je bila oko 18.94%

7 Druga verzija ResNet modela

Za razliku od prethodnog modela koji je radjen isto u ResNet arhitekturu. Odradjeno je par distinktnih promena na modelu. U V1 verziji su svi osim poslednjeg sloja bili zamrznuti sto je znatno ogranicilo sposobnost modela. U ResNet V2 slojevi iz layer4 su ostavljeni za treniranje dok su ostali zamrznuti. Podatci su dodatno sredjeni za laksi rad i omogucava modelu da generalizuje bolje na nepoznatim primerima. U data loaderu je broj radnika povecan na 12 kako bi se ubrzalo ucitavanje podataka kako bi smanjili vreme treninga.

8 Poredjenje rezultata LeNet_v1, ResNet_v1 i ResNet_v2

Poredjenje tacnosti modela:



Epoha	Training loss	Validation Loss
1	0.1365	0.0904
5	0.0368	0.0823
10	0.0195	0.0724
15	0.0130	0.0686
20	0.0089	0.0665

Training loss i validation loss izabranog modela

Kod u ResNet v1 sa pozivom resnet18 i zamenom poslednjeg sloja:

```
model = models.resnet18(pretrained=True)
for param in model.parameters():
    param.requires_grad = False
model.fc = nn.Linear(model.fc.in_features, 3)
model = model.to(device)
```

ResNet v2 kod sa izmenama koje su dovele do bolje tacnost:

```
model = models.resnet18(pretrained=True)
for param in model.parameters():
    param.requires_grad = False
for param in model.layer4.parameters():
    param.requires_grad = True
model.fc = nn.Linear(model.fc.in_features, 3)
model = model.to(device)

scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
    optimizer,
    mode='min',
    factor=0.5,
    patience=2
    )
```