Klasifikacija planeta i satelita

Istraživanje podataka 2
Matematički fakultet

Petra Ignjatović i Todor Todorović
mi20063@alas.matf.bg.ac.rs
mi19241@alas.matf.bg.ac.rs

Sažetak

Ovaj projekat se bavi rešavanjem problema klasifikacije slika koristeći neuronske mreže. Ceo projekat se može naći ovde.

Sadržaj

1 Uvod

Klasifikacija planete i satelita je problem koji se može naći u oblastima astronomije i astrofizike. Za njegovo rešavanje koristite se napredne tehničke klasifikacije pomoću neronskih mreža, što je i bio naš predmet proučavanja.

Cilj projekta je da se, koristeći tehnike obrade slike i mašinskog učenja, predstavi model koji će biti sposoban da prepozna i razlikuje prvo planete, a zatim i njihove prirodne satelite na osnovu njihovih fotografija.

U svrhu istraživanja i procene sposobnosti, modeli napravljeni u ovom projektu su bili podvrgnuti različitim varijantama baze podataka – veličina baze podataka je bila menjana, i same slike na kojima su modeli testirani su bili augmentovani programski alati za različite tehnike.

Prvi model koji će biti predstavljen je imao zadatak da klasifikuje posebnu planetu, posebno satelite, a drugi model je klasifikovao sve objekte zajedno.

2 Pojedinačna klasifikacija

2.1 Neizmenjeni skupovi podataka

Prvi model je radio direktno sa slikama koje je dobio, bez ikakvog njihovog alterniranja.

2.1.1 Klasifikacija planeta

Sunčev sistem se sastoji od 8 planeta, koji ujedno predstavljaju i klase koje će model predviđati: Merkur, Venera, Zemlja, Mars, Jupiter, Saturn, Uran i Neptun.

Baza slika korišćena za njihovu klasifikaciju može se pronaći na sledećem linku.

Svaka planeta ima oko 150 fotografija, što čini osnovni skup za dalju analizu.

U ovom delu projekta, biće prikazana dva načina za učitavanje lične baze podataka, jedan preko `keras.utils` biblioteke, a drugi preko klase `ImageDataGenerator`.

Učitavanje baze podataka

Korišćenjem funkcije `tf.keras.utils.image_dataset_from_directory` neće se izvršiti nikakve izmene nad skupom podataka. Ona jednostavno učitava slike i kreira skup podataka koji se može koristiti za treniranje, testiranje i validaciju modela. Jedini vid preprocesiranja je normalizacija vrednosti piksela - prebacivanje na opseg vrednosti [0,1] - što znači da će gradijenti koji se koriste za ažuriranje težina biti manji i stabilniji, što može dovesti do bržeg i stabilnijeg učenja.

U tensorflow dokumentaciji se mogu naći argumenti koje prima ova funkcija, od kojih su bitniji:

- `directory` - direktorijum u kom se nalaze slike
- `class_names` - eksplicitna lista imena klasa. Ukoliko se ne navede, imena klasa odgovaraju imenima poddirektorijuma
- `batch_size` - broj uzoraka po grupi
- `image_size` - veličina slike
- `validation_split` - mora biti jednak za trening i validacioni skup. Govori koji procenat podataka se čuva za validacioni skup
- `subset` - podskup podataka koji vraća funkcija. Može uzimati vrednost 'training', 'validation' ili 'both'
- `seed` - opciono 'seme' za nasumično mešanje podataka. Koristi se da bi se osiguralo da se pri svakom pokretanju programa podaci mešaju na isti način. Mora biti isti za trening i validacioni skup.

Može se primetiti da ovaj vid učitavanja podataka ne daje test skup, već samo validacioni. Ovo se može prevazići tako što se izvrši dalja podela validacionog skupa - na primer, 10% za validaciju i 20% za test.

Model za klasifikaciju

Opis arhitekture modela:

- `Conv2D` - Ovi slojevi predstavljaju konvolutivne slojeve koji koriste filtere za izdvajanje karakteristika iz slika. U ovom modelu, postoje tri Conv2D sloja sa 128, 256 i 512 filtera, redom. Svaki filter prolazi kroz ulaznu sliku koristeći određeni korak (strides) i izračunava konvoluciju.
- `BatchNormalization` - Ovaj sloj se koristi za normalizaciju izlaza prethodnog sloja. To pomaže u stabilizaciji procesa učenja i ubrzava konvergenciju mreže.
- `SpatialDropout` - Ovi slojevi primenjuju dropout na ulaznom sloju, gde se nasumično određeni neuroni isključuju tokom treninga sa određenom verovatnoćom (u ovom slučaju 20%). Ovo pomaže u prevenciji prenaučenosti i poboljšava generalizaciju modela.
- `MaxPooling2D` - Ovi slojevi primenjuju max pooling operaciju na izlazima prethodnih slojeva kako bi se smanjila dimenzionalnost i izdvojile ključne osobine iz slike.
- `Flatten` - Ovaj sloj se koristi za ravnanje (*flatten*) izlaza iz poslednjeg sloja konvolucije u vektor koji će biti ulaz u potpuno povezane (*dense*) slojeve.
- `Dense` - Ovi slojevi su potpuno povezani i koriste se za klasifikaciju izlaza iz prethodnih slojeva. U ovom modelu, postoji jedan Dense sloj sa 128 neurona i jedan sa 11 neurona (jedan za svaku klasu), koji koristi softmax aktivaciju za generisanje verovatnoća svake klase.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, BatchNormalization, SpatialDropout2D

model = Sequential([
    Conv2D(128, (2,2), strides=(2,2), padding='same', activation='relu', use_bias=True, input_shape=(256, 256, 3)
    BatchNormalization(),
    SpatialDropout2D(0.2),
    MaxPooling2D(pool_size=(2, 2), strides=(2,2), padding='same'),
    Conv2D(256, (2,2), strides=(2,2), padding='same', activation='relu', use_bias=True),
    SpatialDropout2D(0.2),
    MaxPooling2D(pool_size=(2, 2), strides=(2,2), padding='same'),
    Conv2D(512, (2,2), strides=(2,2), padding='same', activation='relu', use_bias=True),
    SpatialDropout2D(0.2),
    MaxPooling2D(pool_size=(2, 2), strides=(2,2), padding='same'),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(11, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Slika 1: Arhitektura modela

Sparse categorical crossentropy je funkcija gubitka koja se koristi za višeklasnu klasifikaciju kada su oznake ciljne klase predstavljene kao celobrojne vrednosti (npr. 0, 1, 2, ...) umesto one-hot kodiranja. Računa gubitak između predviđenih verovatnoća klasa i stvarnih oznaka ciljnih klasa, uzimajući u obzir samo indeks ciljne klase za svaki podatak.

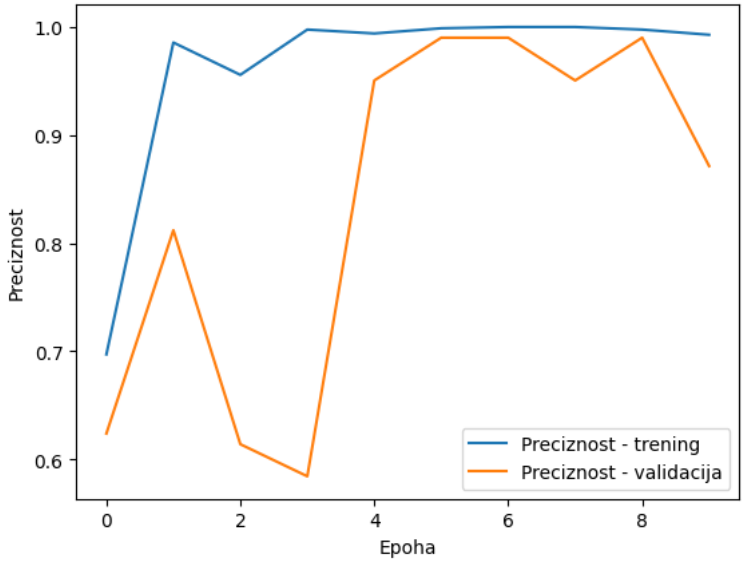
Treniranje i evaluacija modela

U ovom primeru, `EarlyStopping` će pratiti gubitak (*loss*) na validacionom skupu (`monitor='val_loss'`) i zaustaviti trening ako se gubitak ne smanji tokom 3 uzastopne epohe (`patience=3`). Parametar `restore_best_weights=True` će vratiti težine modela na one koje su dale najbolje rezultate na validacionom skupu.

Model pokazuje veliku tačnost na trening skupu, ali nešto manju tačnost na validacionom skupu.

```python
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import EarlyStopping

# Definisanje EarlyStopping callback-a
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

# Treniranje modela sa EarlyStopping callback-om
history = model.fit(train_ds,validation_data=val_ds, epochs=10, callbacks=[early_stopping])

# Prikazivanje grafika tačnosti
plt.plot(history.history['accuracy'], label='Preciznost - trening')
plt.plot(history.history['val_accuracy'], label='Preciznost - validacija')
plt.xlabel('Epoha')
plt.ylabel('Preciznost')
plt.legend()
plt.show()
```

Slika 2: Treniranje modela

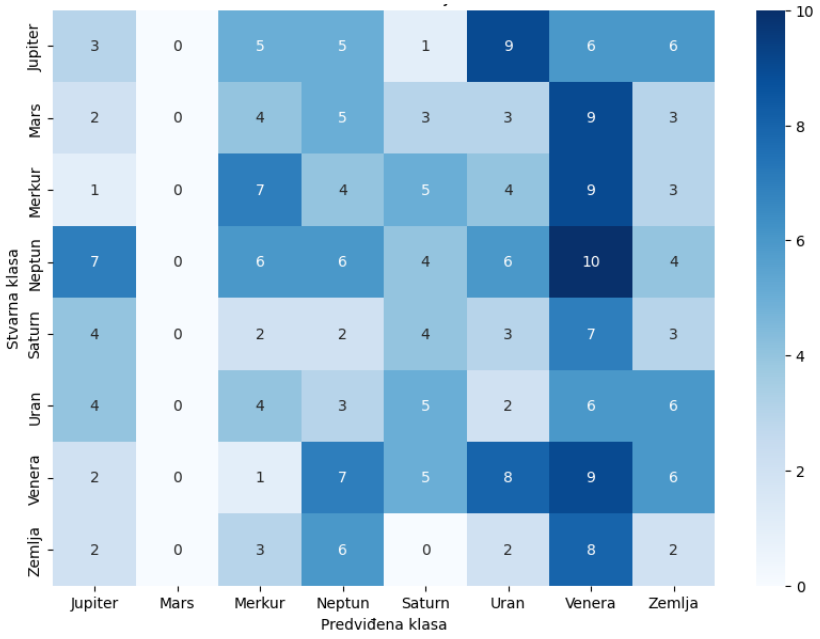


Slika 3: Treniranje modela

Matrica konfuzije

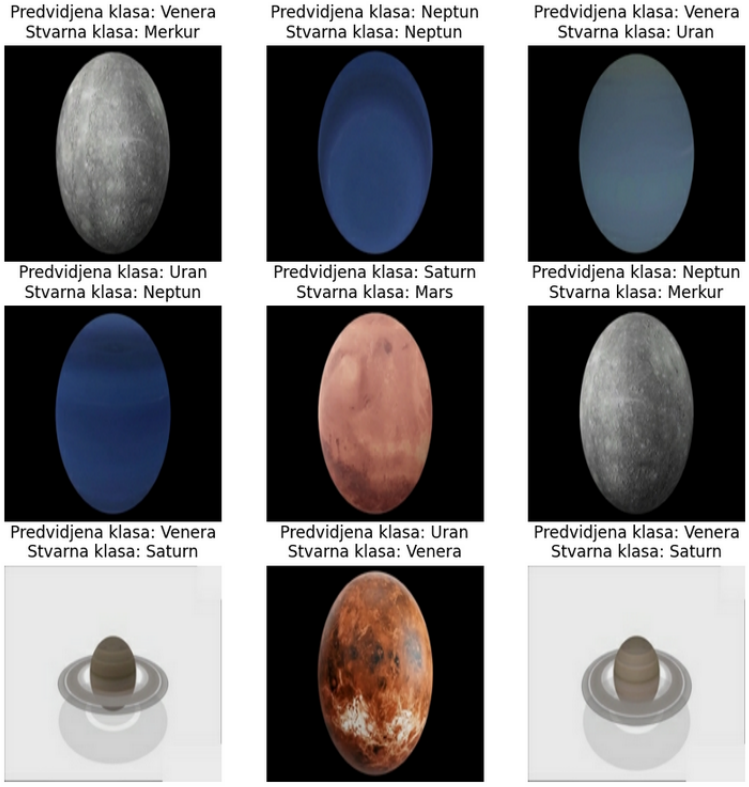
Matrica konfuzije se konstruiše na osnovu predviđanja modela na test skupu. U ovom slučaju, model je najviše uspeha imao u ispravnom predviđanju Venere, ali je isto tako i najviše mešao Neptun i Veneru. Preciznost na test skupu iznosi oko 85%, što je manje u odnosu na trening skup, tako da je to znak da se model preprilagodio.

Na slici br.5 je prikazan rad modela na 9 nasumično odabranih slika iz test skupa.




```
8/8 [==============================] - 2s 237ms/step - loss: 0.7643 - accuracy: 0.8516
Preciznost na test skupu: 0.8515625
Gubitak na test skupu: 0.7642920017242432
```

Slika 4: Matrica konfuzije



Slika 5: Primer upotrebe modela

Zaključak

Model je bio sklon predviđanju klase Venera, i kao anomalija javlja se da nije ni u jednom slučaju predvideo klasu Mars.
Prilikom ponovnog izvršavanja koda, situacija se može ponoviti, ali za različite klase - odnosno, dešava se da model iz nekog razloga ima averziju prema nekoj klasi i ne predviđa je ni za jednu instancu.
S obzirom da su klase balansirane, slike obrađene, i da je ovo veoma specifičan problem, razlog za ovu anomaliju može biti arhitektura računara, i rešenje ovog problema prevazilazi trenutno znanje autora.

Izuzevši to iz obzira, model je pokazao dobre performanse, ali se preprilagodio.

2.1.2 Klasifikacija satelita

Baza slika korišćena za treniranje i evaluaciju modela za klasifikaciju satelita je ručno napravljena skidanjem slika sa zvaničnog Nasinog sajta i preko Google Images.
Pojedini sateliti, na primer, Uranovi i Neptunovi sateliti, su toliko daleko od Zemlje da nisu postojale brojne ekspedicije koje su ih istraživale, i malobrojne dostupne slike su mutne i neupotrebljive za analizu, pa nisu uzete u razmatranje.

Baza se sastoji iz sledećih satelita:

- Zemlja
  - Mesec
- Mars
  - Fobos *(Phobos)*
  - Deimos
- Jupiter
  - Kalisto *(Callisto)*
  - Evropa *(Europa)*
  - Ganimed *(Ganymede)*
  - Io
- Saturn
  - Enkelad *(Enceladus)*
  - Mimant *(Mimant)*
  - Reja *(Rhea)*
  - Titan

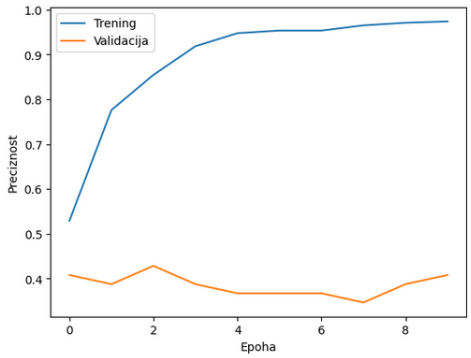
Učitavanje baze

```
data_dir = 'sateliti'

maticne_planete = ['Jupiter','Mars','Saturn','Zemlja']

# Lista satelita
sateliti = ['Evropa', 'Ganimed', 'Io', 'Kalisto', 'Deimos', 'Fobos', 'Enkelad','Mimant','Reja','Titan', 'Mesec']

slike_niz = []  # Bice korisceno za treniranje i evaluaciju modela
slike_putanje = [] # Bice korisceno za prikaz slika iz test skupa
oznake = []
j = 0;

# Popunjavanje slika iz svakog poddirektorijuma
# Svaka kategorija ce biti numerička, označena brojevima 0-10
for i, planeta in enumerate(maticne_planete):
    planeta_dir = os.path.join(data_dir, planeta)
    lista_satelita = os.listdir(planeta_dir)
    for satelit in lista_satelita:
        satelit_dir = os.path.join(planeta_dir,satelit)
        slike = os.listdir(satelit_dir)
        for slika in slike:
            slika_putanja = os.path.join(satelit_dir, slika)
            img = image.load_img(slika_putanja, target_size=(256,256))
            img_array = image.img_to_array(img)
            img_array = preprocess_input(img_array)
            slike_niz.append(img_array)
            slike_putanje.append(slika_putanja)
            oznake.append(j)
        j = j+1
```

Slika 6: Učitavanje satelita preko putanja slika

Treniranje i evaluacija

Konvertovanjem podataka u NumPy nizove se osigurava da su oni u odgovarajućem formatu za dalju obradu u Kerasu ili drugim bibliotekama za mašinsko učenje. Ovo je posebno važno kada se koristi Keras, jer njegove funkcije i metode očekuju da ulazni podaci budu u formi NumPy nizova radi pravilnog funkcionisanja.

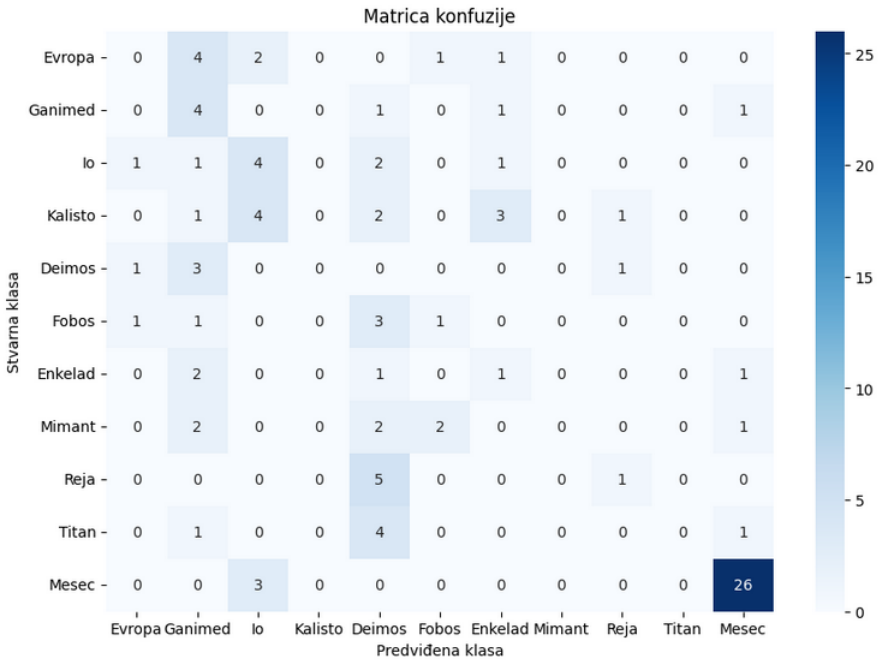


Slika 7: Učitavanje satelita preko putanja slika

Ponovo, model se preprilagodio, a preciznost na validacionom skupu nije rasla.

Matrica konfuzije

Model je imao najviše uspeha u tačnom predviđanju Meseca. Preciznost na test skupu je veoma mala, a gubitak je veliki, što znači da model nema dobru moć generalizacije nad ovim skupom.

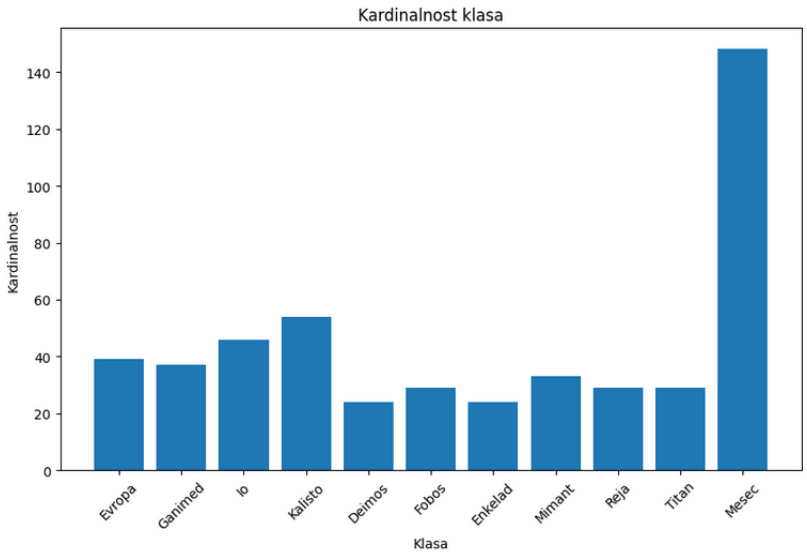


Slika 8: Matrica konfuzije na satelitima

Pre rada sa proširenim skupom podataka, možemo primeniti vid optimizacije - balansiranje klasa.

Prvi vid optimizacije

Klase su poprilično nebalansirane, što se može videti na sledećem grafiku.



Slika 9: Kardinalnost klasa satelita

Da li će balansiranje klasa i uvođenje težina pomoći u optimizaciji modela?

Klasne težine

Argument `class_weight='balanced'` označava da se koristi "balansirana"težina za klase, što znači da će manje zastupljene klase imati veću težinu, a više zastupljene klase manju težinu. Funkcija uzima jedinstvene vrednosti iz niza oznake kao classes argument, koji predstavlja klase koje treba balansirati, dok y=oznake predstavlja sam niz oznaka koje se koriste za izračunavanje težina. Na kraju, funkcija vraća niz težina za svaku klasu, koji se može koristiti u modelu za obučavanje.

```
from sklearn.utils import class_weight

# Izračunanje težine za klase
tezine = class_weight.compute_class_weight(class_weight='balanced', classes=np.unique(oznake), y=oznake)

# Pretvaranje težine u rečnik koji se može koristiti u modelu
tezine_dict = dict(enumerate(tezine))
```

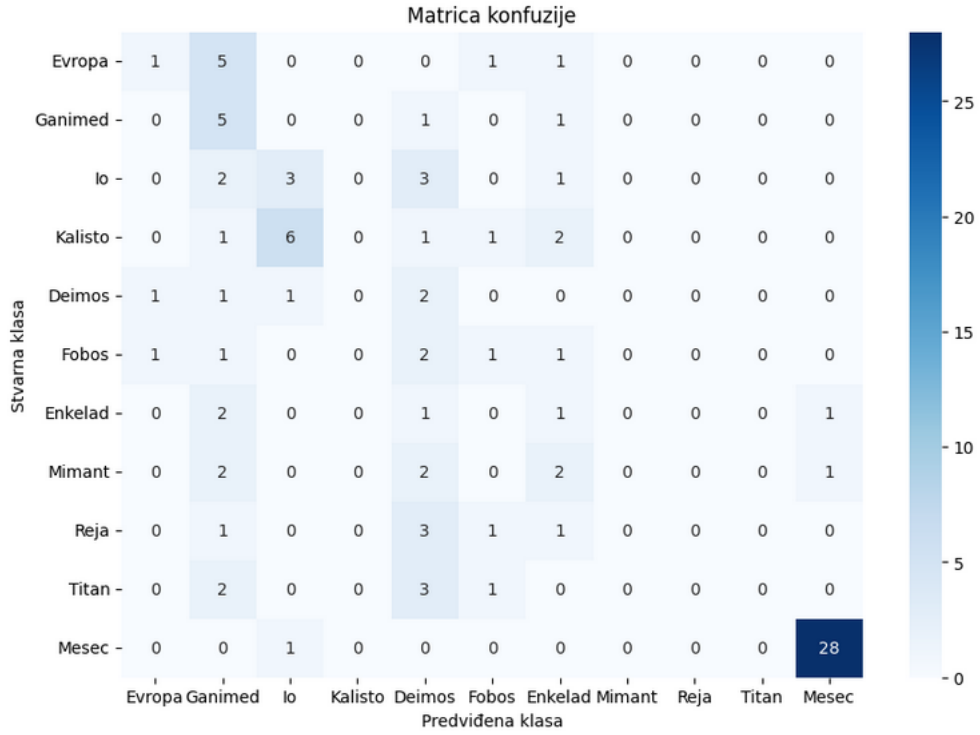
```
# Treniranje modela
history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=10, class_weight=tezine_dict)
```

Slika 10: Upotreba argumenta `class_weight='balanced'`

Da li je model imao poboljšanje posle ove modifikacije će kazati matrica konfuzije.

```python
predikcije = np.argmax(model.predict(np.array(X_test)), axis=-1)

# Stvarne vrednosti za test skup
stvarno = np.array(y_test)

# Izračunavanje matrice konfuzije
matrica_konfuzije = confusion_matrix(stvarno, predikcije)

# Kreiranje DataFrame-a za matricu konfuzije
matrica_df = pd.DataFrame(matrica_konfuzije, index=sateliti, columns=sateliti)

# Prikaz matrice konfuzije
plt.figure(figsize=(10, 7))
sns.heatmap(matrica_df, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predviđena klasa')
plt.ylabel('Stvarna klasa')
plt.title('Matrica konfuzije')
plt.show()

# Evaluacija modela na test skupu
gubitak, preciznost = model.evaluate(np.array(X_test), np.array(y_test))
print("Preciznost na test skupu:", preciznost)
print("Gubitak na test skupu:", gubitak)
```

Slika 11: Primer kreiranja matrice konfuzije

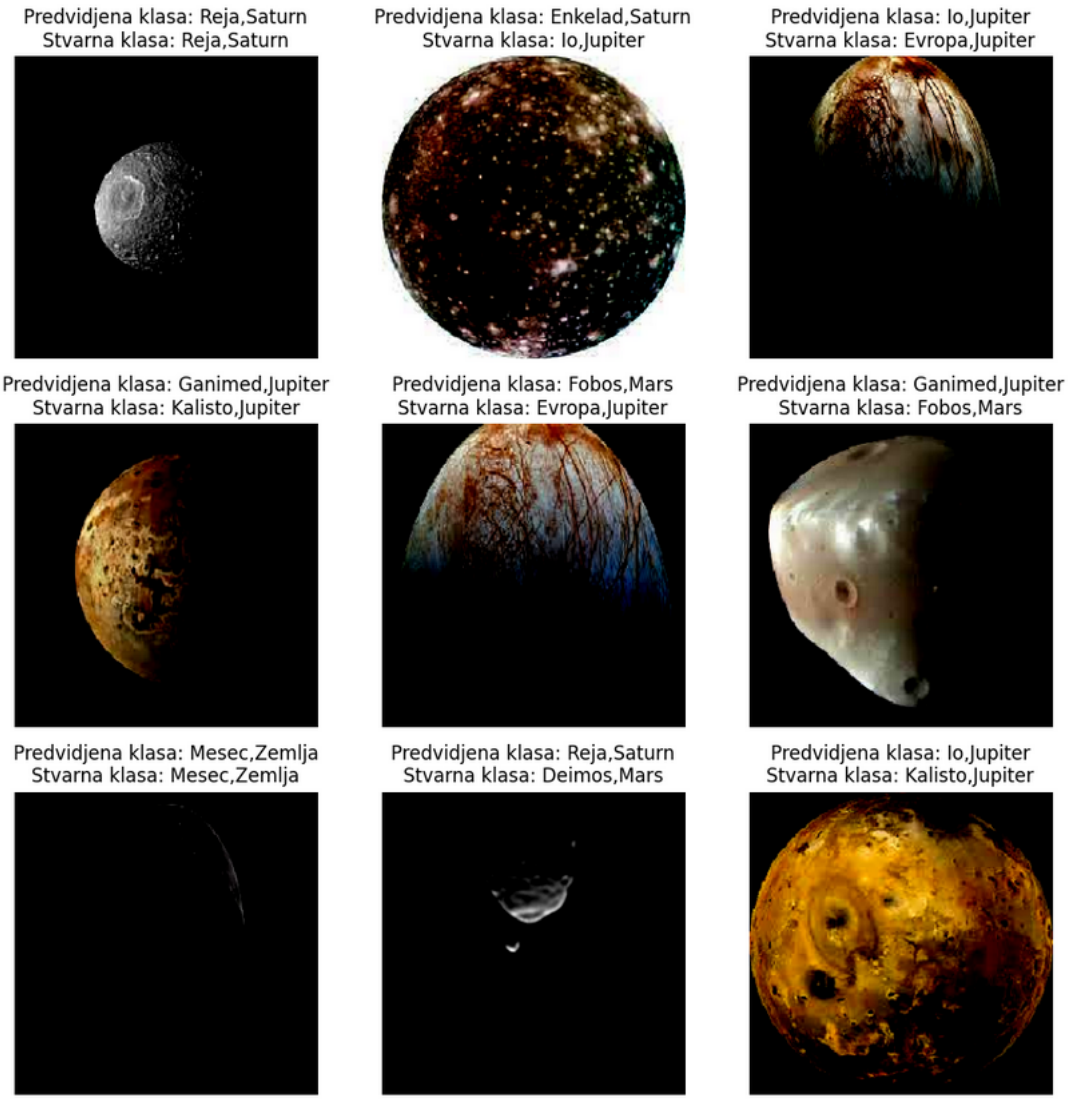



```
4/4 [==============================] - 1s 171ms/step - loss: 2.9908 - accuracy: 0.4141
Preciznost na test skupu: 0.4141414165496826
Gubitak na test skupu: 2.9907708168029785
```

Slika 12: Matrica konfuzije za predviđanje satelita uz balansirane klase

Ovaj metod nije ispunio očekivanja.
Gubitak i preciznost se jesu poboljšali, ali je razlika zanemarljiva.

Ovaj vid optimizacije nije dao zadovoljavajuće rezultate i još uvek je sklon davanju prednosti jednoj klasi, i takođe se preprilagodio - gubitak je veliki. Poboljšanje se može videti u povećanju sposobnosti razlikovanja satelita Io i Zemljinog Meseca. U sledećem delu, biće prikazan drugi mogući način prevazilaženja ovog problema.



Slika 13: Primer upotrebe modela za klasifikaciju satelita

2.2 Augmentovani skupovi podataka

Kao pokušaj poboljšanja efikasnosti modela, drugi vid optimizacije - augmentacija slika može da pomogne. Tako će se baza slika proširiti i model će imati veći skup na raspolaganju za učenje, validaciju i treniranje, i očekuje se da će imati bolju sposobnost prilagođavaja na neviđenim podacima.

Biće primenjen isti model kao i u prvom delu, kako na planetama tako i na satelitima, radi upoređivanja konačnih dobijenih rezultata.

2.2.1 Klasifikacija planeta

Za učitavanje slika, ovaj put će biti prikazano korišćenje klase `ImageDataGenerator`.

Klasa `ImageDataGenerator` u *Kerasu* omogućava različite tehnike augmentacije podataka, poput rotacije, skaliranja, horizontalne i vertikalne refleksije, promene svetline i kontrasta.

`ImageDataGenerator` se obično koristi zajedno sa metodama `flow_from_directory` ili `flow` za generisanje podataka u obliku koji je pogodan za treniranje modela.

Učitavanje baze

I ovog puta, baza slika će biti kreirana pomoću putanja do njih.

```
# Koreni direktorijum u kom se nalaze slike
data_dir = 'planete'

# Definisanje imena klasa u alfabetskom redu
planete = ['Jupiter', 'Mars', 'Merkur', 'Neptun', 'Saturn', 'Uran', 'Venera', 'Zemlja']

# Definisanje praznih listi u koje će biti smeštene putanje i oznake klasa
putanje = []
oznake = []

# Popunjavnje listi
for i, planeta in enumerate(planete):
    planeta_dir = os.path.join(data_dir, planeta)
    fajlovi = os.listdir(planeta_dir)
    putanje += [os.path.join(planeta_dir, fajl) for fajl in fajlovi]
    oznake += [i] * len(fajlovi)
```

Slika 14: Učitavanje baze slika pomoću njihovih putanja

Podela baze na test, trening i validacioni skup

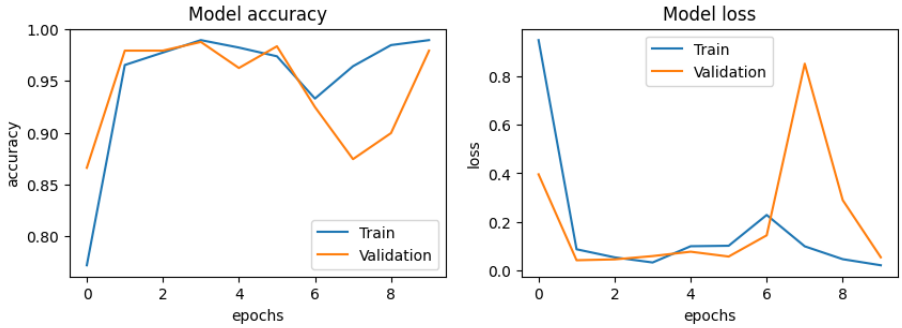
Na slici br.16 se može videti način korišćenja klase `ImageDataGenerator`.
Koristi se `flow_from_dataframe` funkcija koja generiše podatke iz `DataFrame` objekta, gde `filename` predstavlja ime slike, a `class` predstavlja njenu klasu. Parametar `directory` se postavlja na `None` jer su putanje do slika već sadržane u `filename` koloni.
`x_col='filename'` označava da će kolona `'filename'` u DataFrame-u biti korišćena kao ulazni podatak (X) za model, tj. putanja do slika. Slično tome, `y_col='class'` označava da će kolona `'class'` u DataFrame-u biti korišćena kao ciljna promenljiva (Y), tj. oznaka klase kojoj pripada svaka slika. Ovaj pristup omogućava pravilno uparivanje ulaznih slika sa odgovarajućim oznakama klasa tokom procesa učenja modela za klasifikaciju slika.
`target_size` specifizira dimenzije slika koje će biti prosleđene modelu.

`batch_size` određuje broj slika koji će biti prosleđeni modelu u jednoj iteraciji.

`class_mode` se postavlja na `'sparse'` što znači da su labele predstavljene kao 1D nizovi.

Parametar `interpolation` koristi se prilikom promene veličine slika prilikom učitavanja pomoću `ImageDataGenerator` objekta. Interpolacija se koristi za izračunavanje novih vrednosti piksela kada se slika promeni u veličini. U ovom slučaju, `"lanczos"` je metoda interpolacije koja se koristi za glatko skaliranje slika. Lanczos interpolacija je složenija metoda koja pruža bolje rezultate u očuvanju detalja i smanjenju pojave štepenastih"ivica koje se mogu pojaviti kod drugih metoda interpolacije.

Treniranje i evaluacija modela



Slika 15: Grafički prikaz evalucije modela na trening i validacionom skupu

```python
def podeli_bazu(putanje, oznake):

    # Stratifikovana podela skupa na test i trening, 70% za trening, 30% za test
    # Random_state parametar je tu da bi podela pri svakom pokretanju bila istovetna
    X_train, X_test, y_train, y_test = train_test_split(putanje, oznake, test_size=0.3, stratify=oznake, random_state=42)
    # Podela ostatak dataset-a na validacioni (10%) i test (20%) skup
    X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, test_size=2/3, stratify=y_test, random_state=42)

    # Konvertovanje oznaka, koje su trenutno tipa int, u stringove, da bi model mogao da ih primi
    y_train = [str(oznaka) for oznaka in y_train]
    y_test = [str(oznaka) for oznaka in y_test]
    y_val = [str(oznaka) for oznaka in y_val]

    dataframe = pd.DataFrame({'filename':X_train, 'class':y_train})

    # Korišćenje klase ImageDataGeneratori definisanje augmentacija koje će biti primenjene na slike
    datagen = ImageDataGenerator(
        rescale=1./255,  # Normalizacija piksela na [0, 1]
        rotation_range=20,  # Nasumična rotacija slika do 20 stepeni
        width_shift_range=0.2,  # Nasumično pomeranje slika horizontalno do 20% širine
        height_shift_range=0.2,  # Nasumično pomeranje slika vertikalno do 20% visine
        shear_range=0.2,  # Ugao pomeranja u radijanima
        zoom_range=0.2,  # Nasumično uvećanje slika do 20%
        horizontal_flip=True,  # Nasumično horizontalno obrtanje slika
        fill_mode='nearest'  # U slučaju nedostajućih piksela, popuniti vrednosti koristeći najbližu vrednost
    )


    # Generatori za trening, test i validacioni skup
    train_generator = datagen.flow_from_dataframe(
        dataframe=pd.DataFrame({'filename': X_train, 'class': y_train}),
        directory=None,
        x_col='filename',
        y_col='class',
        target_size=target_size,
        batch_size=batch_size,
        class_mode='sparse',
        interpolation="lanczos",
        shuffle=True
    )

    test_generator = datagen.flow_from_dataframe(
        dataframe=pd.DataFrame({'filename': X_test, 'class': y_test}),
        directory=None,
        x_col='filename',
        y_col='class',
        target_size=target_size,
        batch_size=batch_size,
        class_mode='sparse',
        interpolation="lanczos",
        shuffle=False
    )

    val_generator = datagen.flow_from_dataframe(
        dataframe=pd.DataFrame({'filename': X_test, 'class': y_test}),
        directory=None,
        x_col='filename',
        y_col='class',
        target_size=target_size,
        batch_size=batch_size,
        class_mode='sparse',
        interpolation="lanczos",
        shuffle=False
    )

    return train_generator, test_generator, val_generator
```

Slika 16: Upotreba klase `ImageDataGenerator`

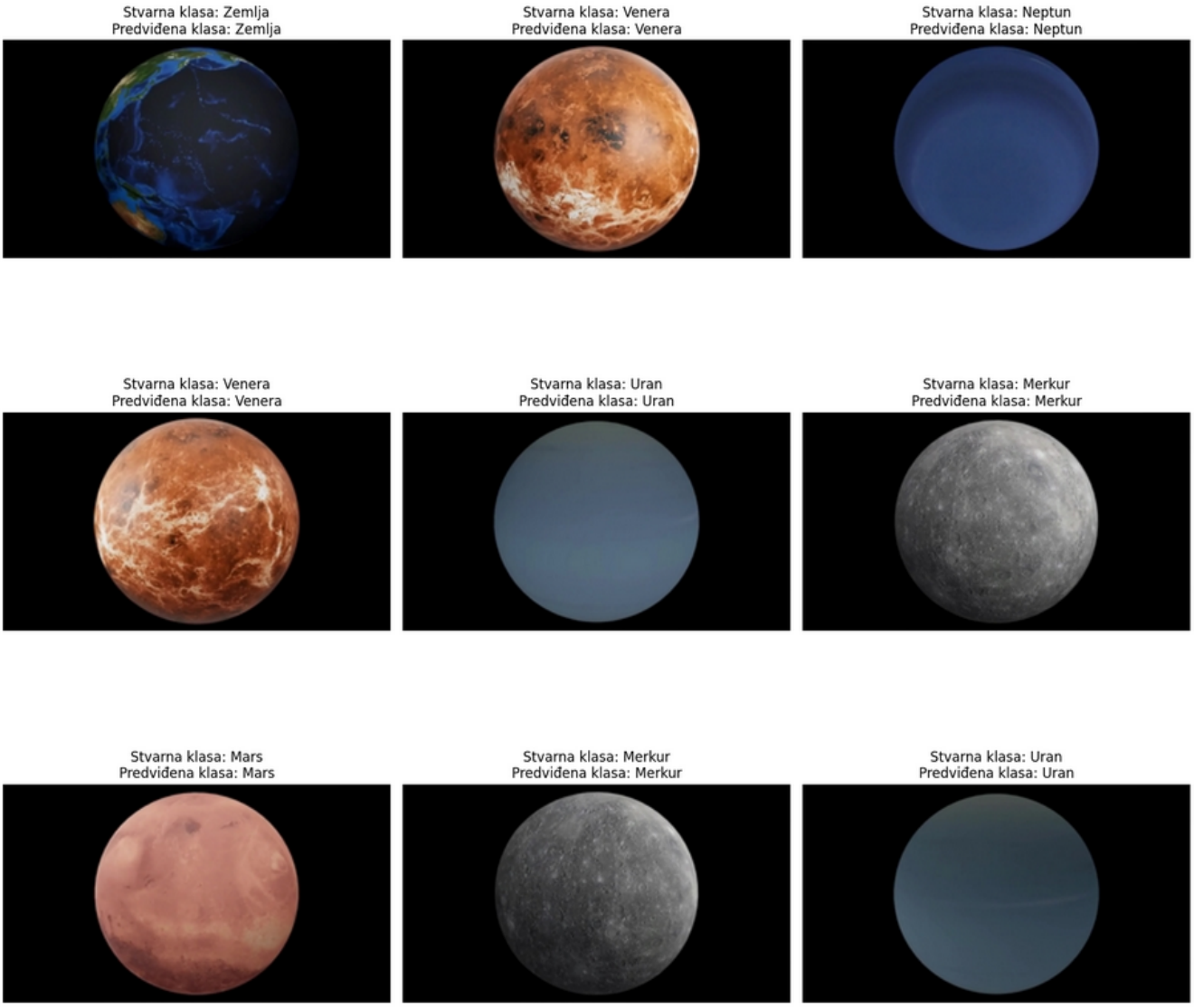
Na grafiku sa slike br. 15 se može uočiti sledeće:

Već posle prve epohe, preciznost na oba skupa je veoma visoka i iznosi oko 97%, uz gubitak od oko 0.01%. Kako su se epohe nastavljale, ove brojke su nastavile da budu poprilično stabilne, gde je preciznost oscilirala između 94% i 97% , uz jednu anomaliju na sedmoj epohi, gde se vidi neobičan pad preciznosti i porast gubitka na validacionom skupu. Međutim, ovaj problem je veoma brzo prevaziđen, i konačan rezultat posle 10 epoha je sledeći:

- `Gubitak: 0.03529`
- `Preciznost: 98.74%`

Upotreba modela

Kao i do sad, primer upotrebe modela je prikazan na 9 nasumično odabranih slika iz test skupa. Slike se biraju iz test skupa jer je potrebno osigurati da ih model nije prethodno već video i naučio, čime se izbegava njegovo kompromitovanje.

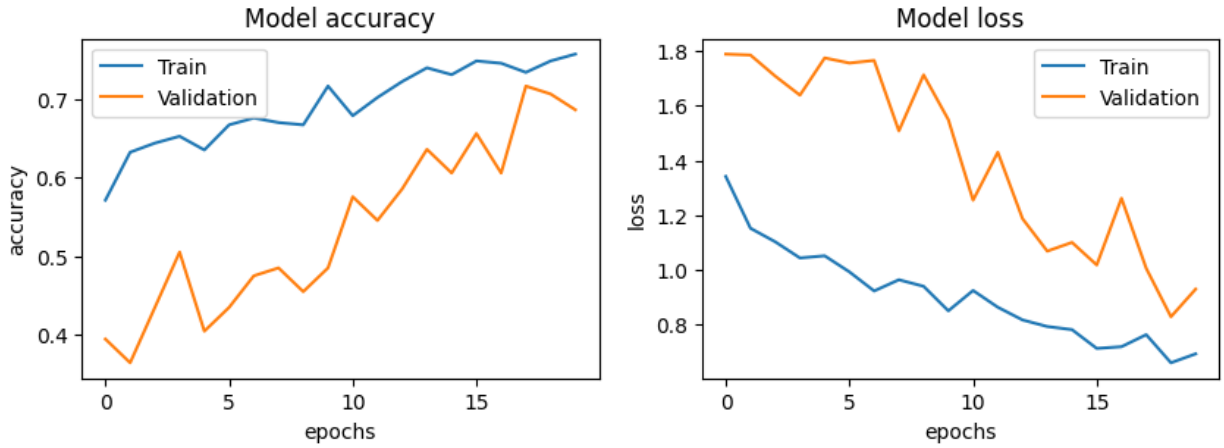


Slika 17: Predikcije modela na skupu planeta

2.2.2 Klasifikacija satelita

Ovaj vid optimizacije je dao zadovoljavajuće rezultate na primeru sa planetama.
Ali, prethodni način je više imao problema sa klasifikacijom satelita, tako da je sledeći izazov videti kako se `ImageDataGenerator` ponaša na ovom skupu.

Grafički prikaz modela na satelitima



Slika 18: Grafički prikaz treniranja modela na satelitima

Prvo što se primećuje, u odnosu na prethodni pristup, je da je preprilagođavanje još uvek prisutno, ali znatno ublaženo - razlika između preciznosti na trening i validacionom skupu je u prvom (neoptimizovanom) pristupu bila oko 40%, a u ovom slučaju je taj broj smanjen na oko 15%. Prava komparacija se može napraviti sa radom modela na test skupu.

```
 # Rezultati na test skupu
rezultati = model.evaluate(test_generator_sateliti, verbose=0)
print("    Gubitak:\033[31m \033[01m {:.5f} \033[30m \033[0m".format(rezultati[0]))
print(" Preciznost:\033[32m \033[01m {:.2f}% \033[30m \033[0m".format(rezultati[1] * 100))
```

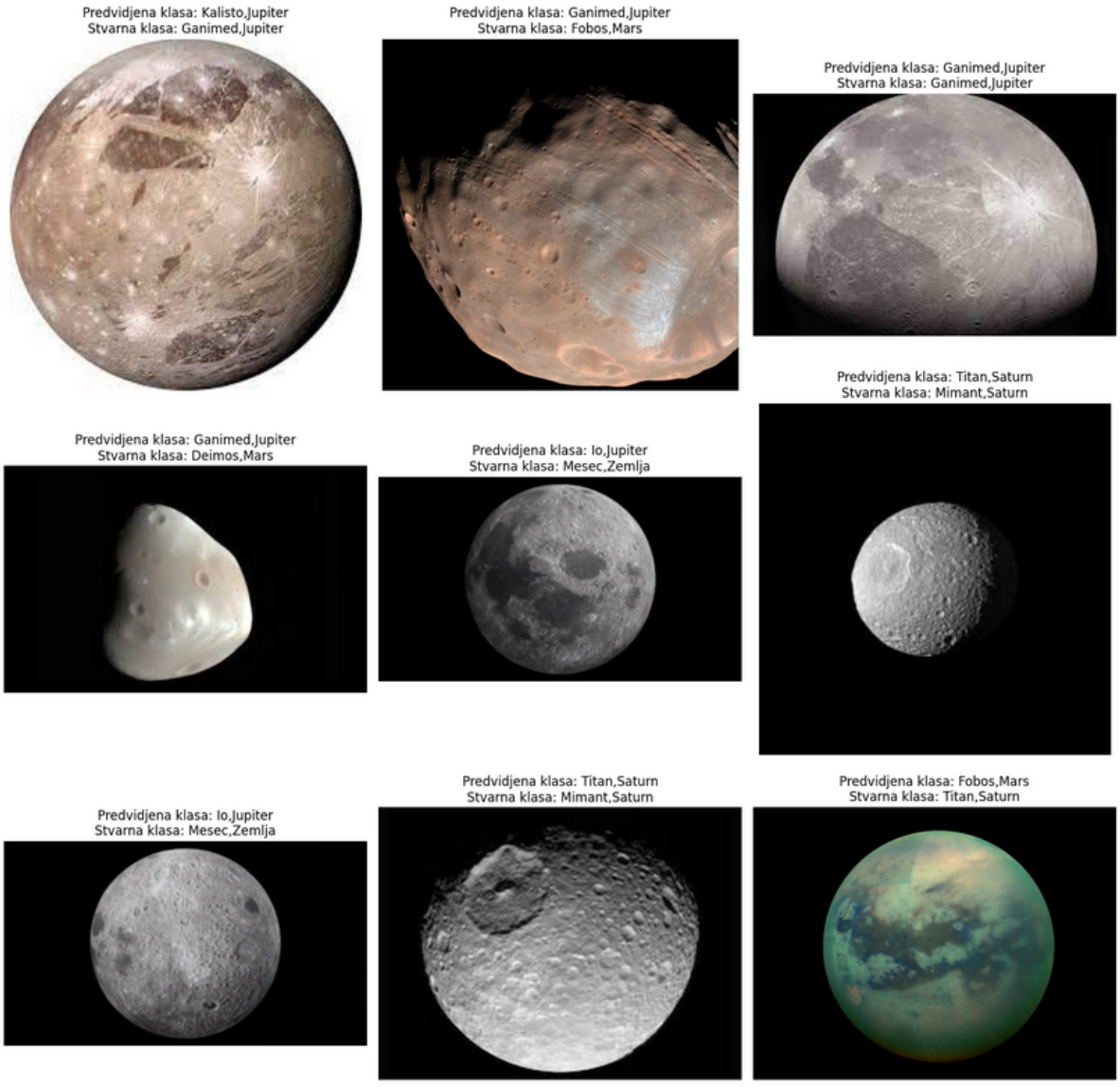
```
   Gubitak:  0.89377
Preciznost:  66.67%
```

Slika 19: Evaluacija modela na test skupu satelita

U odnosu na prethodni (neoptimizovani) slučaj, može se primetiti da je preciznost skoro pa duplo bolja (66.67% u odnosu na 37.37%), i da je gubitak smanjen skoro četiri puta (0.893 u odnosu na 3.37).

Primer upotrebe modela

Na sledećoj strani, na slici br.20, mogu se videti predikcije na devet nasumično odabranih slika iz test skupa satelita.



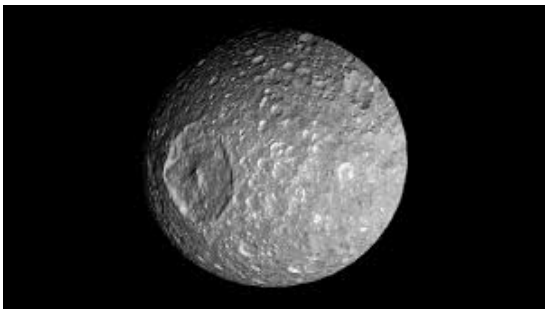
Slika 20: Primer upotrebe modela na test skupu satelita

3 Grupna klasifikacija

U ovoj sekciji, biće predstavljen model čiji je zadatak bio da rukuje i sa planetama i sa satelitima u isto vreme. Znači, broj klasa sa kojima se on suočava je veći. Cilj je videti koliko se mogu optimizovati performanse modela na malo zahtevnijoj bazi podataka koristeći prethodno navedene vidove optimizacije.

Veliki akcenat u ovom radu je bačen na kardinalnost baze kojima će model baratati, ali to jeste veoma bitno u mašinskom učenju iz više razloga.

Kao prvo, ponekad ni sam čovek nije u stanju da diferencira klase koje su pred njim zbog toga što su previše slične. A planete i sateliti, kao nebeska tela, imaju dosta karakteristika koje su im zajedničke. Okrugao oblik je prisutan u svih 19 klasa, koliko ima ova baza, i boje koje se javljaju nisu toliko raznolike. Saturnov Mimant i Zemljin Mesec su odlični primeri gde bi model teoretski imao problema.



Slika 21: Mimant, Saturn

Slika 22: Mesec, Zemlja

Tako da, što je veći broj klasa, veći je izazov za sam model.

Drugi razlog za posebno obraćanje pažnje na kardinalnost baze je što veći broj instanci doprinosi boljoj generalizaciji modela. Sa dovoljno velikim skupom podataka, model ima veću šansu da nauči relevantne uzorke za svaku klasu i da se bolje nosi sa raznolikošću podataka.

3.1 Neizmenjeni skupovi podataka

```
Epoch 1/10
37/37 [==============================] - 33s 857ms/step - loss: 1.2703 - accuracy: 0.6243 - val_loss: 123.1237 - v
al_accuracy: 0.6793
Epoch 2/10
37/37 [==============================] - 32s 855ms/step - loss: 0.4827 - accuracy: 0.8346 - val_loss: 75.3124 - va
l_accuracy: 0.7772
Epoch 3/10
37/37 [==============================] - 33s 872ms/step - loss: 0.3045 - accuracy: 0.8889 - val_loss: 102.0982 - v
al_accuracy: 0.8098
Epoch 4/10
37/37 [==============================] - 33s 882ms/step - loss: 0.2165 - accuracy: 0.9271 - val_loss: 179.5439 - v
al_accuracy: 0.7826
Epoch 5/10
37/37 [==============================] - 32s 840ms/step - loss: 0.1092 - accuracy: 0.9712 - val_loss: 189.8067 - v
al_accuracy: 0.7826
Epoch 6/10
37/37 [==============================] - 33s 875ms/step - loss: 0.0844 - accuracy: 0.9796 - val_loss: 347.8977 - v
al_accuracy: 0.7880
Epoch 7/10
37/37 [==============================] - 33s 877ms/step - loss: 0.0784 - accuracy: 0.9737 - val_loss: 192.9258 - v
al_accuracy: 0.8152
Epoch 8/10
37/37 [==============================] - 32s 850ms/step - loss: 0.0439 - accuracy: 0.9856 - val_loss: 313.9784 - v
al_accuracy: 0.7663
Epoch 9/10
37/37 [==============================] - 35s 936ms/step - loss: 0.0305 - accuracy: 0.9881 - val_loss: 225.0860 - v
al_accuracy: 0.7880
Epoch 10/10
37/37 [==============================] - 34s 906ms/step - loss: 0.0246 - accuracy: 0.9924 - val_loss: 219.2405 - v
al_accuracy: 0.7826
```

Slika 23: Treniranje modela kroz 10 epoha

Treniranje modela po epohama u mašinskom učenju i dubokom učenju podrazumeva iterativno poboljšanje modela kroz prolazak kroz ceo skup podataka više puta. Svaka epoha predstavlja jedan prolazak kroz skup podataka, tokom kojeg model ažurira svoje parametre kako bi smanjio gubitak i poboljšao svoje predikcije. Ovaj proces omogućava modelu da nauči složene obrasce u podacima i postane sve bolji u rešavanju zadatka za koji je dizajniran.

Na slici br. 23 je prikazano kako se model ponašao kroz 10 epoha treniranja.

Razlika između preciznosti na trening i validacionom skupu je značajna, što znači da je došlo do preprilagođavanja, a pokazatelj na to je i velika vrednost funkcije gubitka.

```
# Evaluacija modela na test skupu
gubitak, preciznost = model.evaluate(test_ds)
print("Preciznost na test skupu:", preciznost)
print("Gubitak na test skupu:", gubitak)
```

```
10/10 [==============================] - 2s 143ms/step - loss: 257.4950 - accuracy: 0.7812
Preciznost na test skupu: 0.78125
Gubitak na test skupu: 257.4949645996094
```

Slika 24: Evaluacija modela na zajedničkom test skupu, verzija 1

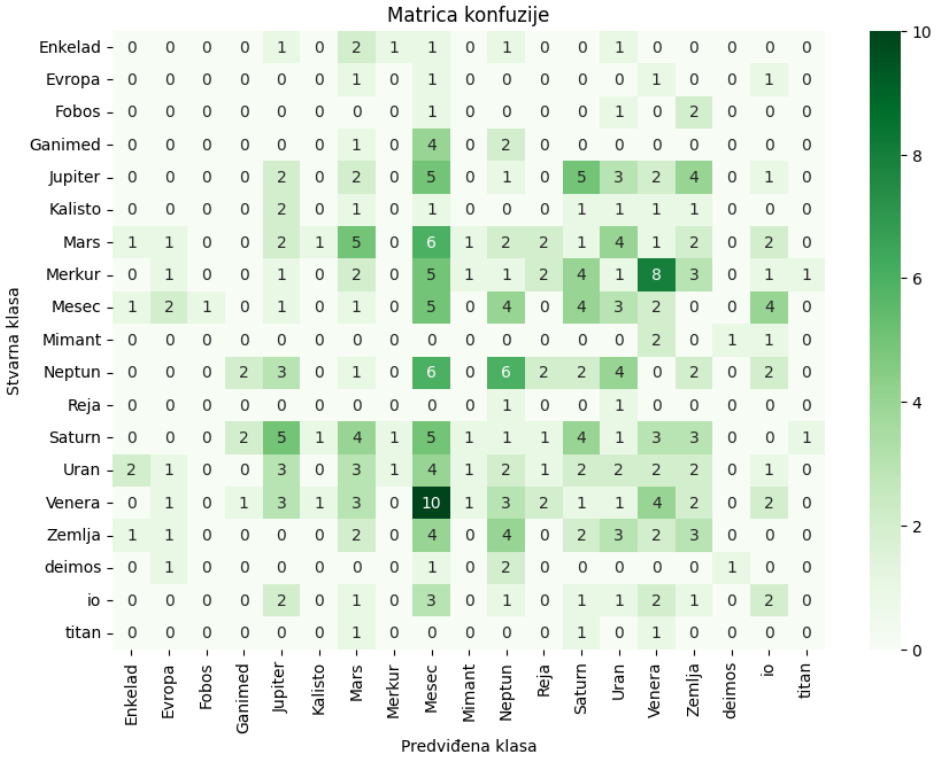
Na osnovu test skupa možemo samo potvrditi taj zaključak, gubitak je nedopustivo veliki.

Razlog za to može biti nedovoljna kompleksnost modela, i kao pokušaj uklanjanja tog uzroka, u model je dodat još jedan `dropout` sloj za regularizaciju, i testiranje je ponovljeno.

```
10/10 [==============================] - 4s 431ms/step - loss: 141.7634 - accuracy: 0.7906
Preciznost na test skupu: 0.7906249761581421
Gubitak na test skupu: 141.7633819580078
```

Slika 25: Evaluacija modela na zajedničkom test skupu, verzija 2

Dakle, samo jedan jedini sloj, iako nije uspeo previše da poveća preciznost, uspeo je da vidno smanji gubitak. Iako to još uvek nije zadovoljavajuća mera, napredak je primetan.



Slika 26: Matrica konfuzije, kombinovane klase

Sa matrice konfuzije, može se pročitati sledeće:

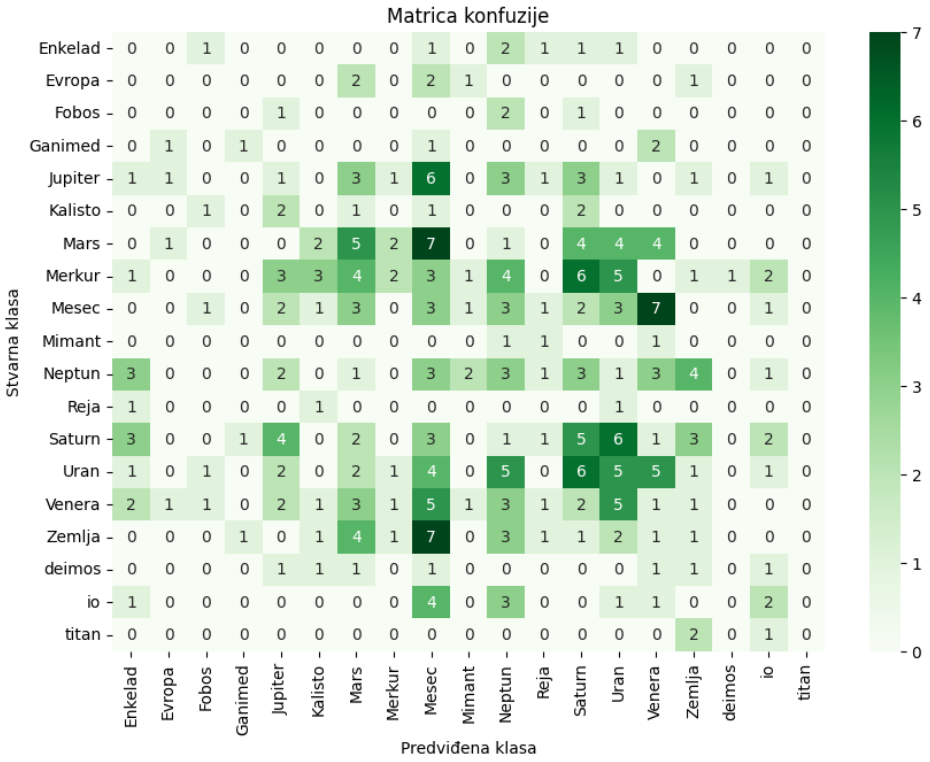
- Model je često za Veneru mislio da je Mesec
- Model je često za Merkur mislio da je Venera
- Model je imao teškoća u klasifikovanju satelita uopšte, većina kolona koje pripadaju satelitima su popunjene nulama i jedinicama

Može se naći opravdanje za nedovoljnu obučenost modela u razlikovanju dve, i na oko veoma slične, planete, ali nešto se mora uraditi za povećanje naklonosti u klasifikaciji samih satelita.

Inicijalna ideja bi mogla da bude balansiranje klasa; međutim, kako to nije dalo dobre rezultate u prvom delu ovog izveštaja, nema dovoljno opravdanja da to bude pokušaj rešavanja problema i u ovom delu. Tako da ćemo odmah preći na oružje za koje je dokazano da deluje, a to je postupak augmentacije slika.

3.2 Augmentovani skupovi podataka

Očekivanja nakon korišćenja klase `ImageDataGenerator` su da model ima smanjenju funkciju gubitka, smanjeno učenje napamet i bolju sposobnost prilagođavanja.



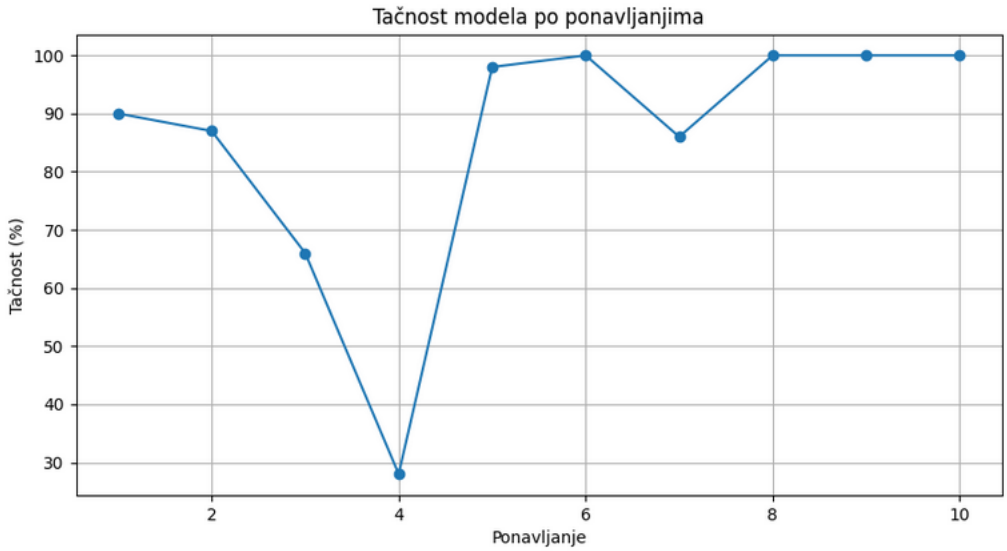
Slika 27: Matrica konfuzije, kombinovane klase

Vizuelno gledano, još uvek se mešaju iste klase, i još uvek je pojačana pristranost prema klasifikaciji planeta, ali je afinitet prema satelitima iole poboljšan.

Još jedan način vizualizacije mogućnosti ovog modela je kroz ponovljene testove.

1. U glavnoj petlji, za svaki pokušaj, nasumično se biraju slike iz testnog skupa.
   Parametar `replace=False` u funkciji `np.random.choice` označava da se elementi ne smeju birati više puta, odnosno da svaki indeks može biti izabran samo jednom. Ovo je korisno ako želimo da izbegnemo ponavljanje istih slika u testnom skupu kako bismo dobili nepristranu procenu performansi modela.
2. Nakon toga, za svaku odabrana sliku, model se koristi za predikciju njenog sadržaja.
3. Ako je predikcija tačna (tj. indeks predviđene klase je jednak pravoj oznaci), brojač tačnih predikcija se povećava.
4. Na kraju, tačnost za taj pokušaj se računa kao odnos tačnih predikcija i ukupnog broja predikcija, i dodaje se u listu tačnosti.
5. Nakon što se izvrše svi pokušaji, lista će sadržati tačnosti za svaki pokušaj testiranja.

Dakle, u 10 epizoda, primenjujemo model na 100 nasumično odabranih slika iz test skupa, bez ponavljanja.



Slika 28: Grafik uspešnosti modela po ponavljanjima

Ponovo se javlja anomalija u jednom primeru. Na četvrtoj epizodi je ostvarena tačnost manja od 30% što je daleko manje u poređenju sa ostalim epizodama. Medijana iznosi 94%, a prosek iznosi 86.5%, Ovo ukazuje na to da većina predvidjanja modela ima visoku tačnost, s obzirom na to da je medijana blizu maksimalne vrednosti od 100%. Međutim, postoji nekoliko manje tačnih predviđanja (ispod 94%) koja su uticala na prosečnu vrednost. Ovo može značiti da model dobro funkcioniše u većini slučajeva, ali može imati poteškoća sa specifičnim primerima.

I ovog puta je postignuto unapređenje modela koristeći ovaj način optimizacije i veštačkog proširenja skupa.

4 Zaključak

4.1 Pojedinačna klasifikacija

Kada se fokusiramo na pojedinačno predviđanje planeta i satelita bez dodatne augmentacije podataka, primećujemo značajne razlike u performansama. Dok model pokazuje visoku preciznost u klasifikaciji planeta, sa minimalnim gubicima na test skupu, situacija kod satelita nije tako ružičasta. Model se suočava sa izazovima u identifikaciji klasa, što može biti rezultat nedovoljne raznolikosti u slikama, nebalansiranih klasa ili nedovoljno velike baze podataka. Balansiranje klasa nije davalo očekivane rezultate, ali introdukcija augmentiranih slika značajno poboljšava performanse modela, posebno u kontekstu klasifikacije satelita. Ovo sugeriše da dodatna raznolikost koju donose augmentirane slike omogućava modelu da bolje generalizuje i efikasnije klasifikuje satelite.

4.2 Grupna klasifikacija

Kada se odlučimo da modeliramo zajedno planete i satelite bez augmentacije podataka, primećujemo visoku preciznost na test skupu, ali uz ozbiljne izazove. Model pokazuje tendenciju ka preprilagođavanju, uz ogroman gubitak i potencijalno lošije performanse na neviđenim podacima. Uvođenje augmentirane baze podataka kao strategije za prevazilaženje problema uočenih u prethodnom pristupu donosi značajno poboljšanje. Gubici se smanjuju, dok se sposobnost modela da se generalizuje na nove podatke značajno unapređuje. Ipak, mogu se pojaviti novi izazovi vezani za preprilagođavanje modela na augmentirane podatke.

Kroz sve ove testove, primetili smo razlike u performansama modela i karakteristikama predikcija. Ovi rezultati ukazuju na važnost pažljivog pristupa u izboru metoda obrade podataka i predikcije, u zavisnosti od specifičnih zahteva zadatka. Za dalji napredak, preporučuje se dublja analiza uticaja različitih faktora na performanse modela i njihovog optimalnog kombinovanja radi postizanja što preciznijih predikcija planeta i satelita na osnovu dostupnih slika.