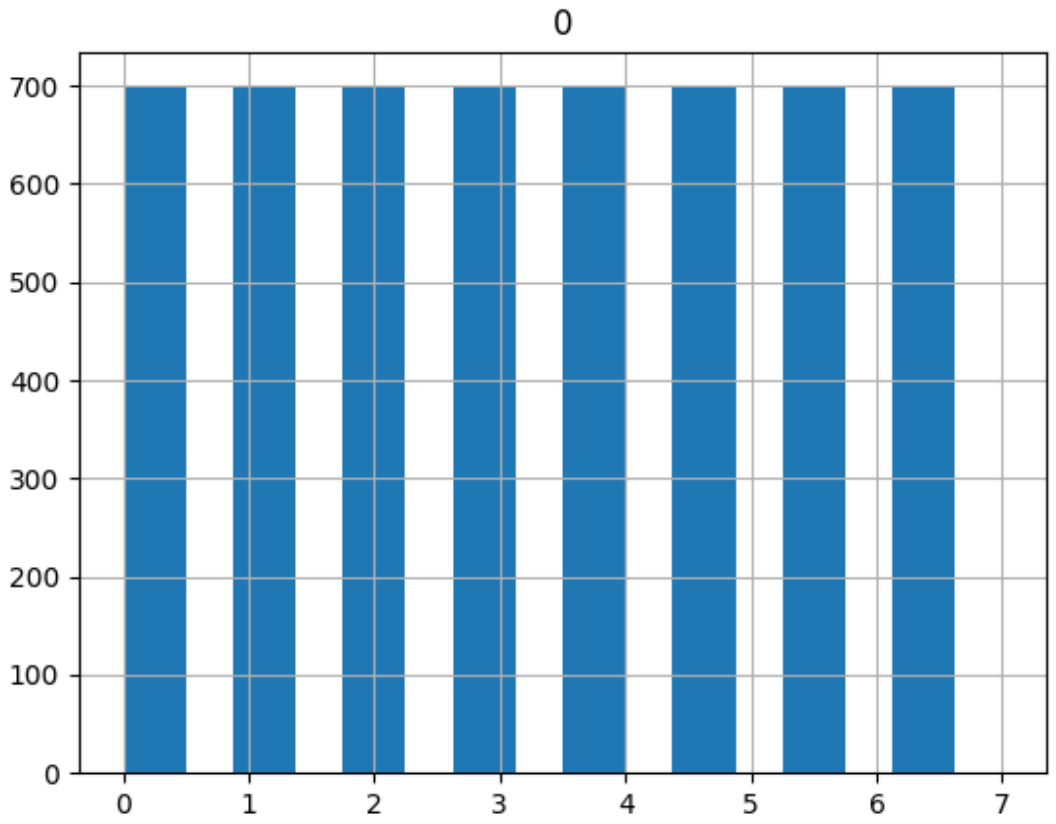
Izveštaj

Duboko učenje

Opis dataset-a

Problem koji se rešava nad dataset-om je klasifikacija biljaka. Dataset se sastoji od osam klasa (kasava, kukuruz, patlidžan, pomorandža, ljuta paprika, soja, spanać, lubenica) i svaka klasa se sastoji od 800 slika. Trening podaci se nalaze u folderu „projekat_dataset" u kome je za svaku klasu napravljen folder sa imenom te klase, koji sadrži tačno 700 slika primeraka klase. Test podaci se nalaze u odvojenom folderu „test" u kome je za svaku klasu napravljen folder sa imenom te klase, koji sadrži tačno 100 slika primeraka klase.

Broj primeraka svake klase u trening skupu



Histogram pokazuje da svaka klasa ima tačno 700 primeraka

Pošto je jednak broj primeraka svake klase nije potrebno primeniti nijednu od tehnika za rešavanje problema nebalansiranih klasa (undersampling, oversampling, class weights).

```
14
15  #Pravljenje histograma za prikaz primera
16
17
18  Y= pd.read_csv( filepath_or_buffer: "podaciB.csv",header=None)
19
20
21  plt.figure()
22  Y.hist(width=0.5,bins=8)
23  plt.show()
24
25
26
```

Kod uz pomoć koga je utvrđeno koliko svaka klasa ima primeraka

Prikaz jednog primerka svake klase

Po jedan primerak svake klase dat je na slici ispod kao i kod kojim su prikazani, za koj je napravljen novi folder „primer“, čija je struktura ista kao i za folder „projekat_dataset”, samo što je u svakom folderu umesto 700 slika stavljena po jedna slika koja predstavlja primer te klase.


```
38
39  from keras.utils import image_dataset_from_directory
40
41  N = 8
42
43  Xprimer = image_dataset_from_directory( directory: 'primer/',
44                                          image_size=img_size, shuffle=0)
45
46  classes = Xprimer.class_names
47
48  print(classes)
49
50
51  plt.figure()
52  for img, lab in Xprimer.take(1):
53      for i in range(N):
54          plt.subplot( *args: 4, int(N/4), i+1)
55          plt.imshow(img[i].numpy().astype('uint8'))
56          plt.title(classes[lab[i]])
57          plt.axis('off')
58
59  plt.show()
60
```

Slike primeraka i kod uz pomoć koga su predstavljeni ti primerci

Podela podataka na odgovarajuće skupove

Podela podataka na trening i validacioni skup je bitna kako bi se u toku treniranja performanse modela proveravale na skupu podataka koji ne utiče na promenu težina u modelu i time sprečila mreža da „napamet“ nauči trening podatke, tj. svaki odbirak iz trening skupa ispravno klasifikovala, ali zato izgubila na generalizaciji (preobučavanje). Trening skup podataka se deli na trening i na validacioni skup, sa tim da je trening skup 80% podatak, dok je validacioni skup 20% ulaznih podataka. Podaci za trening i validacioni skup su nasumično izmešani kako model ne bi mogao da se „navikne” na odredjene podatke, već je podacima promenjen redosled da ne bi išlo recimo 700 slika pomorandži, pa onda 700 slika lubenice, jer bi se onda model „navikao“ na pomorandže, pa na lubenice itd.

Test podaci su potpuno nezavisni i oni su učitani iz posebnog foldera „test“. Oni su bitni kako bi se performanse obučenog modela utvrdile na skupu koji se uopšte nije koristio u procesu treniranja.

```
52  Xtrain = image_dataset_from_directory(main_path,
53                                        subset='training',
54                                        validation_split=0.2,
55                                        image_size=img_size,
56                                        batch_size=batch_size,
57                                        seed=123)
58
59  Xval = image_dataset_from_directory(main_path,
60                                      subset='validation',
61                                      validation_split=0.2,
62                                      image_size=img_size,
63                                      batch_size=batch_size,
64                                      seed=123)
65
66  Xtest = image_dataset_from_directory(test_path,
67                                       image_size=img_size,
68                                       batch_size=batch_size,
69                                       seed=123)
```

Učitavanje i podela podataka na trening, validacioni i test skup

Uz pomoć opcije seed rečeno je kakvo je početno stanje od koga treba krenuti nasumično birati primerke za ova dva skupa, argument image_size prima tuple sa dimenzijama slike, batch_size predstavlja broj podataka koji se zajedno grupišu, subset koji je tip skupa u pitanju, a validation_split koliki udeo podataka se odvaja za validaciju.

Predprocesiranje podataka

Za predprocesiranje podataka koristila se tehnika skaliranja (layers.Rescaling) koja menja opseg vrednosti piksela sa originalnog celobrojnog opsega [0, 255] na realni opseg [0, 1].

Osim skaliranja izvšena je i augmentacija podataka, koja menja ulazne podatke tako što ih okreće, rotira i zumira.

Preobučavanje i zaštita od preobučavanja

Preobučavanje je pojava gde mreža gubi svoju sposobnost generalizacije podataka, tako da podatke nad kojima je mreža trenirana može sa velikom tačnošću da klasifikuje, dok ostale podatke sa kojima nije imala do tada susret mnogo lošije klasifikuje, tako da je tačnost na test skupu mnogo lošija od tačnosti na trening skupu. Preobučavanje može da se desi ukoliko je model previše kompleksan za ulazne podatke kojima se obučava ili ukoliko se predugo obučava pa „napamet nauči“ podatke.

Korišćene tehnike za sprečavanje preobučavanja su rano zaustavljanje, L2 regularizacija i dropout.

Rano zaustavljanje sprečava model da se predugo podučava nad trening podacima tako što se obučavanje prekida kada se utvrdi da izabrana metrika u određenom broju uzastopnih epoha nije prešla preko ranije dobijene vrednosti. Zatim se restauriraraju težine iz te epohe koja je dala najbolju vrednost metrike. Korišćena metrika je tačnost predikcije validacionog skupa, a obučavanje se zaustavlja nakon 25 lošijih epoha.

Regularizacija pomaže da model ne postane previše osetljiv na trening podatke i poboljšava njegove performanse na nepoznatim podacima, tako što „kažnjava“ velike vrednosti težina i time sprečava da male promene u ulaznim podacima dovode do velikih promena na izlazu. Ovo se postiže dodavanjem još jednog člana (zavisnog od težina) kriterijumskoj funkciji. Konkretna tehnika regularizacije koja je korišćena je L2 regularizacija sa vrednošću 0.001 koeficijenta regularizacije (lambda) i ona je izabrana zato što može da nauči kompleksnije oblike podataka (kao što su slike). Kod nje se kriterijumskoj funkciji dodaje zbir kvadrata težina.

$$L2 : L(\omega) = E(\omega) + \lambda \cdot \sum(\omega^2)$$

Dropout tehnika nasumično „isključuje“ odredjen procenat neurona u svakoj epohi. Ovime se poboljšava generalizacija, tako što se model primorava da nauči opštije karakteristike i da se ne oslanja previše na dominantne karakteristike odnosno putanje u mreži.

Formiranje i obučavanje neuralne mreže

```
data_augmentation = Sequential([
        layers.RandomFlip("horizontal", input_shape=(img_size[0], img_size[1], 3)),
        layers.RandomRotation(0.25),
        layers.RandomZoom(0.1),
])

model = Sequential([
    data_augmentation,
    layers.Rescaling(1. / 255, input_shape=(img_size[0], img_size[1], 3)),
    layers.Conv2D(16, 3, padding='same', activation='relu', kernel_regularizer=l2(0.001)),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu', kernel_regularizer=l2(0.001)),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu', kernel_regularizer=l2(0.001)),
    layers.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(128, activation='relu', kernel_regularizer=l2(0.001)),
    layers.Dense(num_classes, activation='softmax')
])

model.summary()

model.compile(Adam(learning_rate=0.001),
              loss=SparseCategoricalCrossentropy(),
              metrics='accuracy')

es = EarlyStopping(monitor='val_accuracy', patience=25, restore_best_weights=True)

history = model.fit(Xtrain,
                    epochs=100,
                    validation_data=Xval,
                    callbacks=[es],
                    verbose=1)
```

Formiranje i obučavanje neuralne mreže

Izabrana kriterijumska funkcija je Sparse Categorical Crossentropy, zato što ona predstavlja najbolju opciju za n-arnu klasifikaciju. Takođe, ona se koristi kada su labele celobrojnog tipa, a ne dobijene one-hot encoding-om.

Za metodu optimizacije kriterijumske funkcije se koristi Adamov optimizator, zato što rešava razne probleme koji se javljaju kod drugih metoda optimizacije. Adamov optimizator ima mogućnost da modifikuje konstantu učenja i zbog toga performanse modela manje zavise od izbora hiperparametra. Pokazuje dobre performanse i pri radu sa visokodimenzionalnim parametarskim prostorima što je slučaj kod klasifikaciji slika.

Aktivaciona funkcija neurona u skrivenim slojevima mreže je relu. Izabrana je zato što rešava probleme poput nestajucih gradijenata, koj se javlja kod sigmoid i tanh funkcija. U izlaznom sloju se koristi softmax funkcija, zato što se ulazni podaci klasifikuju na veći broj klasa.

Koristi se sekvencijalna arhitektura neuralne mreže u kojoj se ulazni podaci najpre predprocesiraju na ranije navedene načine. Zatim nailaze na niz 2D konvolucionih i puling slojeva kao što je prikazano na slici iznad. Pri kreiranju konvolucionih slojeva prvi parametar predstavlja broj izlaznih filtera, drugi dimenzije konvolucionog filtera, a padding=’same’ znači da se podaci dopunju nulama kako ne bi došlo do gubitka informacija. Dropout se primenjuje ubacivanjem layers.Dropout sloja. Potpuno povezani slojevi (npr. Dense) očekuju ravni vektor kao ulaz, pa se zato koristi layers.Flatten() da bi se prešlo sa četvorodimenzionalnih mapa osobina (batch_size, visina, širina, dubina) na dvodimenzionalni vektor (batch_size, visina * širina * dubina). Na kraju slede dva potpuno povezana sloja, od kojih prvi ima 128 neurona, a drugi 8 neurona, zato što je to izlazni sloj i vrši se klasifikacija na 8 klasa.

Broj parametara u konvolucionim slojevima se dobija sledećom formulom:

$$(visinaFiltera * sirinaFiltera * brojUlaza + 1) * brojIzlaznihFiltera$$

Broj parametara u potpuno povezanim slojevima se dobija sledećom formulom:

$$(brojUlaza * brojNeurona) + brojNeurona$$

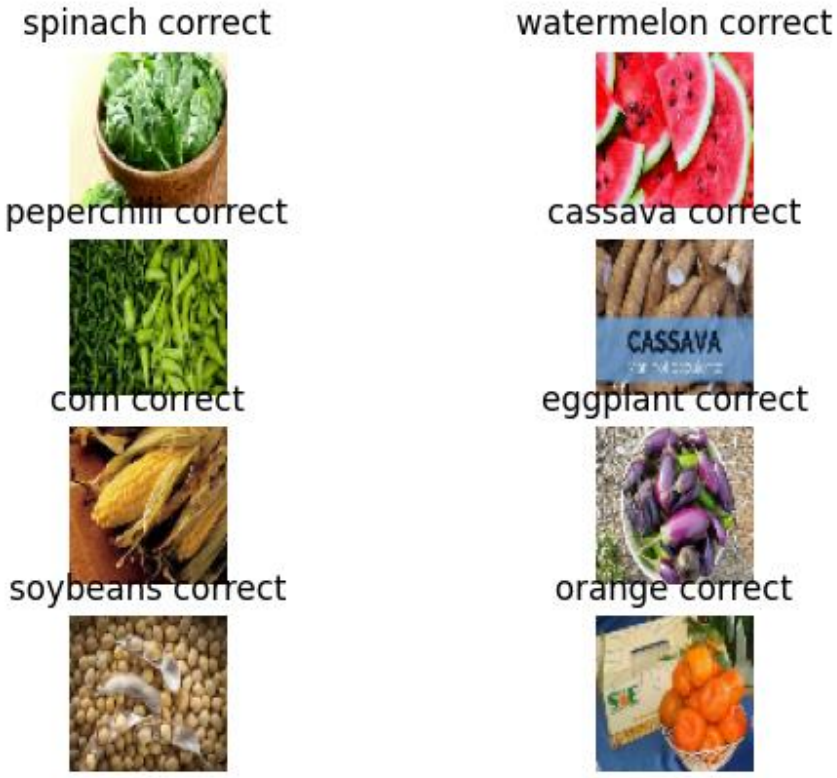
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 64, 64, 3)         0

 rescaling (Rescaling)       (None, 64, 64, 3)         0

 conv2d (Conv2D)             (None, 64, 64, 16)        448

 max_pooling2d (MaxPooling2  (None, 32, 32, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        4640

 max_pooling2d_1 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 dropout (Dropout)           (None, 16, 16, 64)        0

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 128)               2097280

 dense_1 (Dense)             (None, 8)                 1032

=================================================================
Total params: 2121896 (8.09 MB)
Trainable params: 2121896 (8.09 MB)
Non-trainable params: 0 (0.00 Byte)
```

Ukupan broj parametara i broj parametara po slojevima

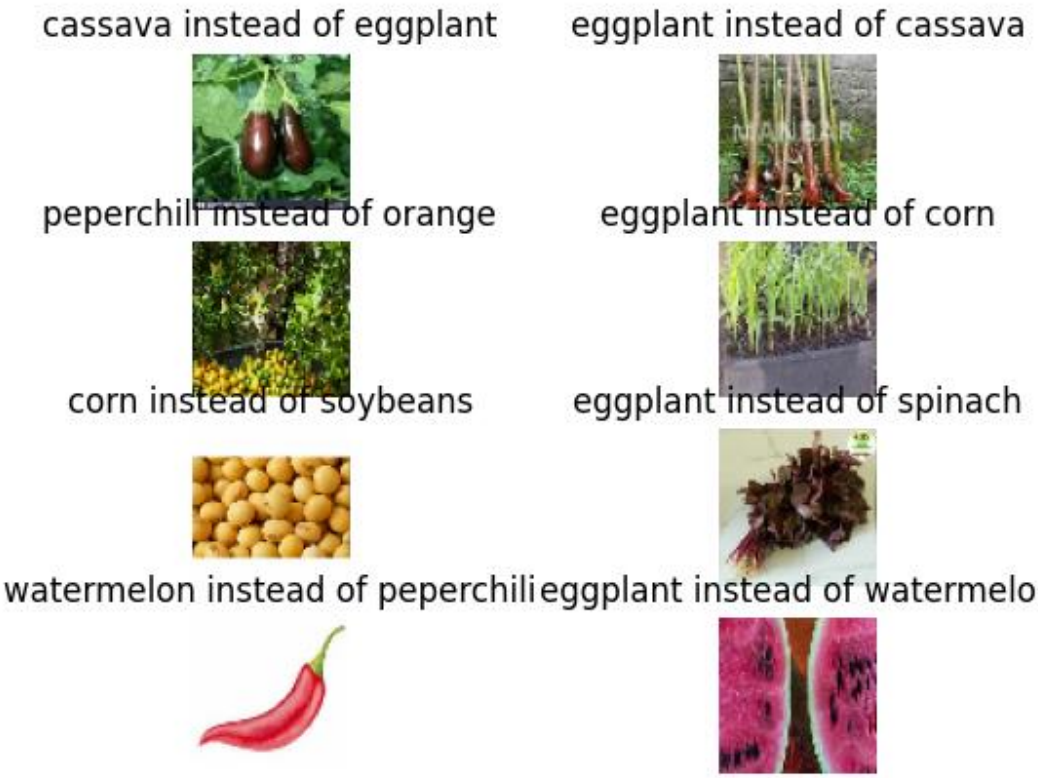
Predviđanje i prikaz performansi

```
135  correct = {"cassava": True, "corn": True, "eggplant": True, "orange": True, "peperchili": True,
136             "soybeans": True, "spinach": True, "watermelon": True}
137  incorrect = {"cassava": True, "corn": True, "eggplant": True, "orange": True, "peperchili": True,
138               "soybeans": True, "spinach": True, "watermelon": True}
139  correct_img = []
140  correct_lab = []
141  incorrect_img = []
142  incorrect_lab = []
143  incorrect_pred = []
144
145  labels = np.array([])
146  pred = np.array([])
147
148  for img, lab in Xtest:
149      prediction = np.argmax(model.predict(img, verbose=0), axis=1)
150      labels = np.append(labels, lab)
151      pred = np.append(pred, prediction)
152
153      if not (any(correct) or any(incorrect)):
154          continue
155      for i in range(len(prediction)):
156          if prediction[i] == lab[i] and correct[classes[lab[i]]]:
157              correct[classes[lab[i]]] = False
158              correct_img.append(img[i])
159              correct_lab.append(lab[i])
160          elif prediction[i] != lab[i] and incorrect[classes[lab[i]]]:
161              incorrect[classes[lab[i]]] = False
162              incorrect_img.append(img[i])
163              incorrect_lab.append(lab[i])
164              incorrect_pred.append(prediction[i])
165
166  print('Tačnost modela je: ' + str(100 * accuracy_score(labels, pred)) + '%')
```

Predviđanje i izdvajanje dobro i loše klasifikovanih primeraka. Dobijena tačnost je 76.6%

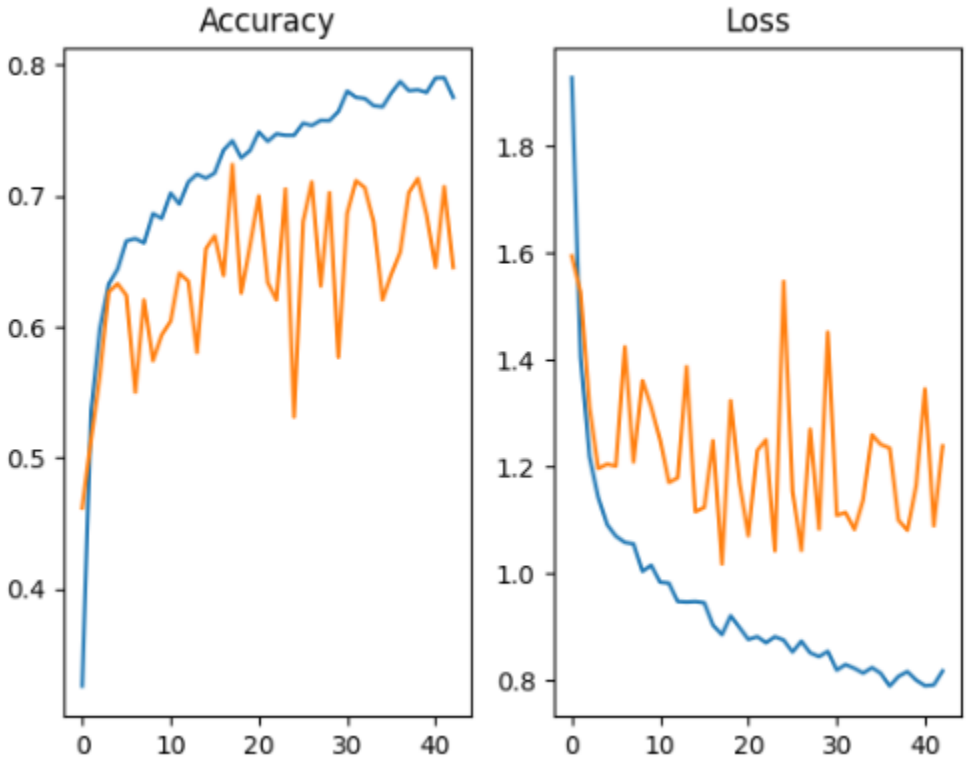


Dobro raspoređeni primerci



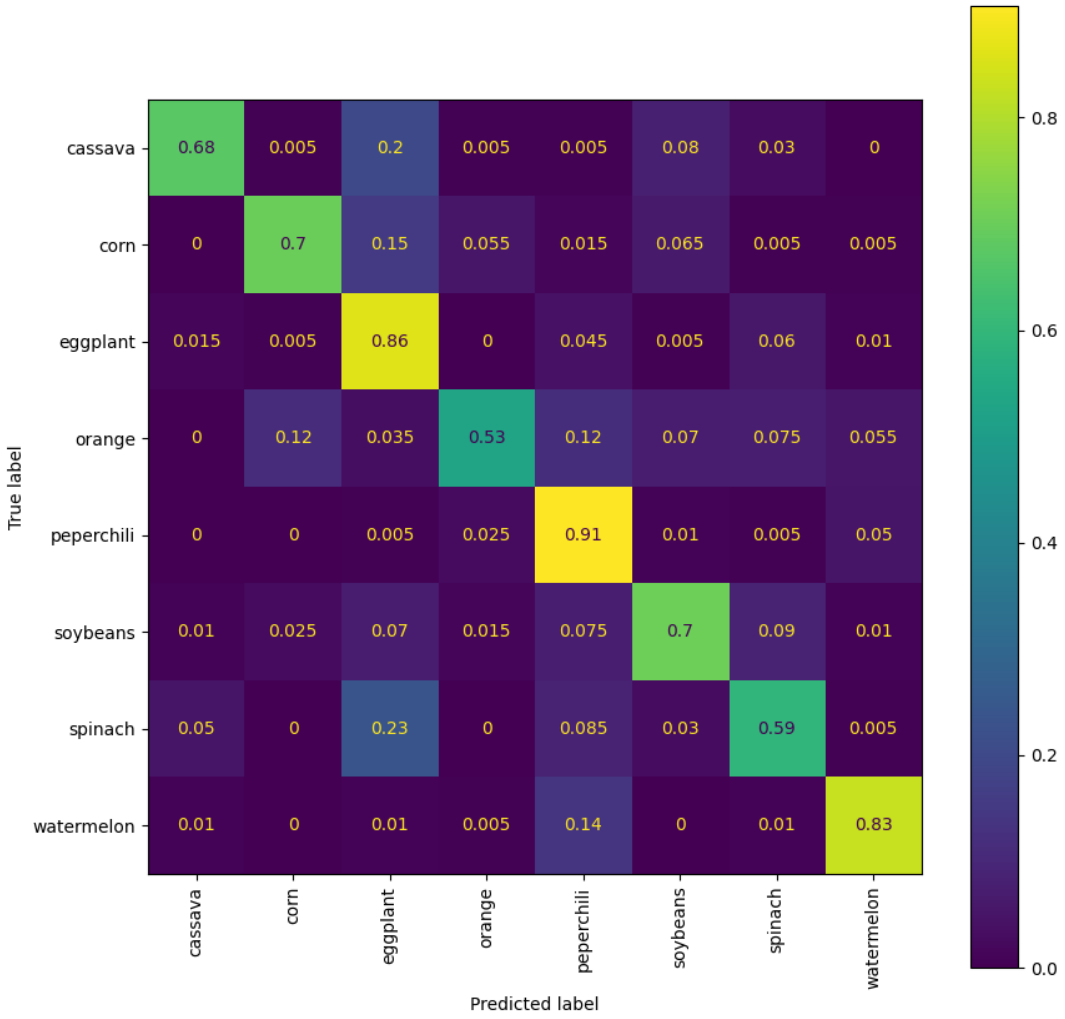
Loše raspoređeni primerci

```python
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure()
plt.subplot(121)
plt.plot(acc)
plt.plot(val_acc)
plt.title('Accuracy')
plt.subplot(122)
plt.plot(loss)
plt.plot(val_loss)
plt.title('Loss')
plt.show()
```

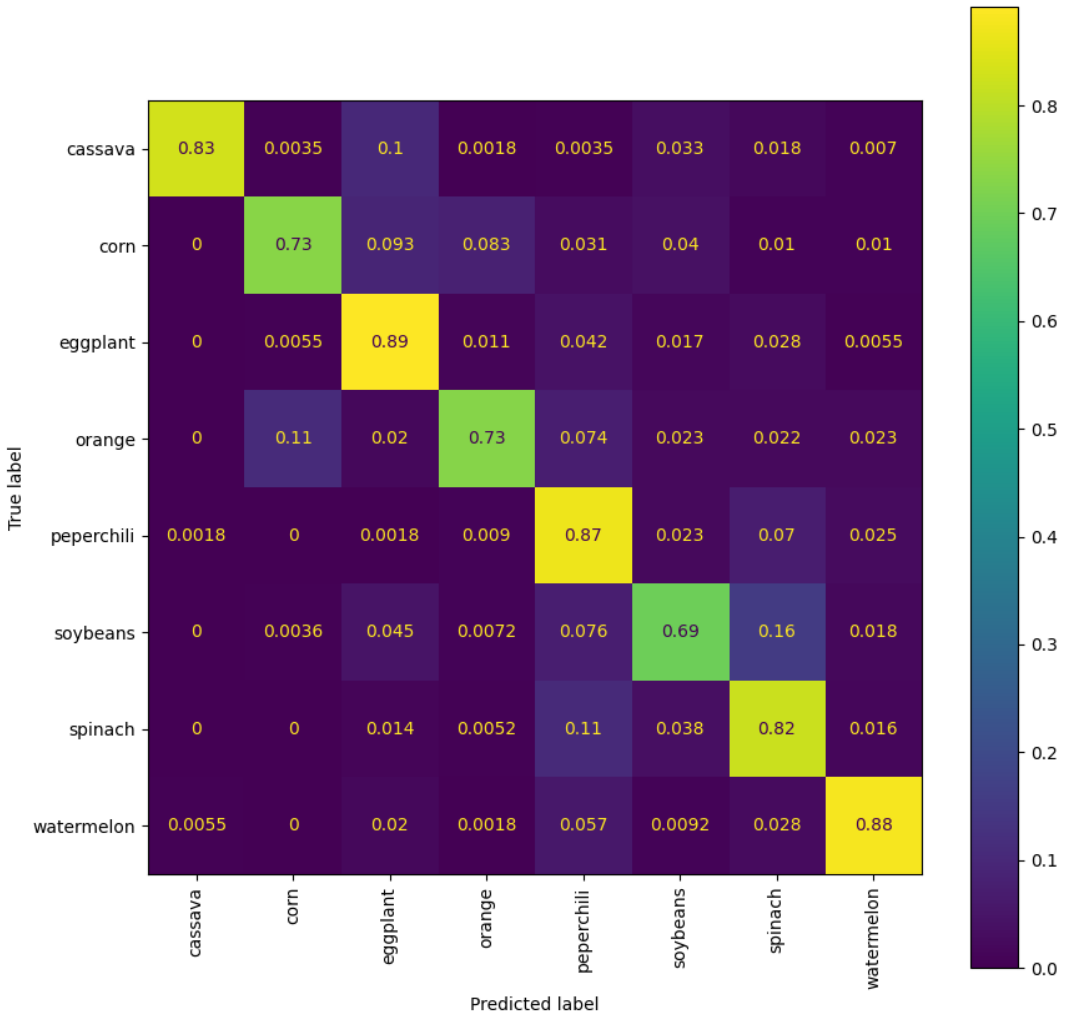


Grafik performanse neuralne mreže kroz epohe obučavanja nad trening i validacionom skupu

```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(labels, pred, normalize='true')
cmDisplay = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
cmDisplay.plot()
plt.show()
```



Konfuziona matrica za test skup



Konfuziona matrica za trening skup

Ilija Miletić 2021/0335

Petar Milojević 2021/0336