# Modeliranje i simulacija neuronskih mreža koristeći Matlab

**Ružman, Lovro**



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNALSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Sveučilišni prijediplomski studij Računalstvo**

# MODELIRANJE I SIMULACIJA NEURONSKIH MREŽA KORISTEĆI MATLAB

**Završni rad**

**Lovro Ružman**

**Osijek, 2024.**

# SADRŽAJ

# 1. UVOD

Neuronske mreže, poznate kao i umjetne neuronske mreže, su računalni sustavi koji su inspirirani ljudskim mozgom. Koriste se za rješavanje različitih problema u strojnom učenju, obradi prirodnog jezika, računalnom vidu i drugim područjima. Sastoje se od međusobno povezanih umjetnih neurona koji obrađuju i prenose informacije. Svaki neuron prima ulazne signale, te ih obrađuje, zatim prenosi informacije. Neuroni međusobno komuniciraju kroz izmjenu signala. Izlazni signali jednog neurona postaju ulazni signali za druge neurone, stvarajući složenu mrežu koja može učiti i pamtiti informacije.

Zadatak ovog rada je pružiti uvid u ovu tehnologiju implementiranu kroz MATLAB. U sljedećem poglavlju obrađuju se svi alati koji se mogu koristiti osim MATLAB-a, također prednosti i mane koje ti alati imaju. Nadalje, u trećem poglavlju dan je osnovni primjer koji se koristi kao objašnjenje neuronskih mreža. Zatim se u četvrtom se poglavlju prolaze kompleksniji primjeri rada s mrežama koje rješavaju problem klasifikacije i raspoznavanja uzoraka. U petom poglavlju je napravljen zadatak za laboratorijsku vježbu koji je prilagođen jednom terminu vježbi. Taj zadatak je napravljen kombinacijom kompleksnijih primjera iz četvrtog poglavlja. Konačno, posljednje poglavlje u kojemu je zaključen cijeli rad.

## 1.1. Zadatak završnog rada

Zadatak ovog završnog rada je napraviti laboratorijsku vježbu koristeći tehnologije za izradu neuronskih mreža u MATLAB-u. Potrebno je proći kroz osnovni primjer kroz koji se nauče osnove neuronskih mreža, zatim napraviti i objasniti nekoliko kompleksnijih primjera i od njih izraditi vježbu.

# 2. PREGLED PODRUČJA TEME

Neuronske mreže predstavljaju jedan od ključnih alata u području strojnog učenja, s brojnim primjenama u klasifikaciji, predviđanju, prepoznavanju obrazaca i optimizaciji. MATLAB, kao jedan od vodećih softverskih alata za inženjering i znanstvene proračune, pruža moćne mogućnosti za razvoj, treniranje i analizu neuronskih mreža putem svojih specijaliziranih alata, poput Neural Network Toolboxa (sada poznatog kao Deep Learning Toolbox) [1].

## 2.1. Uvod u upotrebu MATLAB-a za neuronske mreže

MATLAB je postao popularan izbor za istraživače i inženjere koji rade s neuronskim mrežama zbog svoje integracije s drugim matematičkim i inženjerskim alatima, intuitivnog korisničkog sučelja i opsežne dokumentacije. Deep Learning Toolbox omogućava korisnicima izgradnju i treniranje različitih tipova neuronskih mreža, uključujući potpuno povezane mreže, konvolucijske mreže (CNN) i rekurentne mreže (RNN).

## 2.2. Aktualne primjene i metode u MATLAB-u

U kontekstu primjena neuronskih mreža u MATLAB-u, najčešće metode uključuju klasifikaciju slika, vremenske serije, analizu podataka i modeliranje predviđanja. Primjerice, konvolucijske neuronske mreže (CNN) se često koriste za prepoznavanje i klasifikaciju slika u medicinskim i industrijskim aplikacijama, dok su LSTM (Long Short-Term Memory) mreže popularne za analizu vremenskih serija i predviđanja u financijskom sektoru.

Klasifikacija slika: MATLAB-ov Deep Learning Toolbox omogućava jednostavno treniranje CNN modela s unaprijed podešenim mrežama kao što su AlexNet, VGG-16 i ResNet. Ove mreže se često koriste u zadacima kao što su klasifikacija objekata i prepoznavanje lica.

Analiza vremenskih serija: Rekurentne mreže i LSTM modeli omogućavaju analizu i predviđanje vremenskih serija, što je korisno u područjima kao što su financijska tržišta, analiza potražnje i predviđanje vremenskih prilika. MATLAB podržava treniranje i evaluaciju ovih modela s prilagodljivim parametrima, što omogućava optimizaciju performansi.

Optimizacija i kontrola procesa: Neuronske mreže se koriste za optimizaciju složenih inženjerskih procesa, kao što su kontrola robota, optimizacija industrijskih procesa i predviđanje kvarova. MATLAB-ove funkcije za optimizaciju i simulaciju omogućavaju simulaciju i testiranje različitih scenarija uz primjenu neuronskih mreža.

## 2.3. Usporedba sa drugim alatima i metodama

Iako MATLAB pruža snažan alat za rad s neuronskim mrežama, postoji nekoliko drugih platformi i biblioteka koje se često koriste u iste svrhe. Na primjer, Python sa svojim bibliotekama TensorFlow i PyTorch nudi veću fleksibilnost i brže izvođenje zbog podrške za GPU akceleraciju i dinamičko računanje grafa. S druge strane, MATLAB nudi bolju integraciju s drugim inženjerskim alatima i jednostavniju vizualizaciju podataka, što ga čini idealnim za prototipiranje i obrazovne svrhe [2] [3].

# 3. JEDNOSTAVAN PRIMJER IZ MATLAB-A

Ovaj primjer služi kao uvod u osnovne koncepte neuronskih mreža, koristeći unaprijed pripremljene i obučene modele. Cilj je pružiti početnicima jasno razumijevanje rada neuronskih mreža kroz praktične primjere (Slika 3.1) [4].

Prvo se učitava slika pomoću funkcije „*imread*“.

Funkcija „*detectTextCRAFT*“ pronalazi tekstove na slici pomoću modela dubinskog učenja prepoznavanja regije znakova za detekciju teksta (CRAFT). Ta funkcija koristi prethodno obučeni CRAFT model dubokog učenja za otkrivanje teksta na slici. Prethodno obučeni CRAFT model može otkriti 9 jezika koji uključuju kineski, japanski, korejski, talijanski, engleski, francuski, arapski, njemački i hindski (indijski).

Pomoću funkcije „*insertShape*“ se nacrtaju granični okrivi na slici.

Zatim sam prikaz teksta na slici, koji se dobije funkcijom „*imshow*“.

```
I = imread("big_ferit.jpg");

bboxes = detectTextCRAFT(I);

Iout = insertShape(I,"rectangle",bboxes,LineWidth=3);

figure

imshow(Iout)
```

*Slika 3.1. MATLAB implementacija jednostavnog primjera*

Kao što je prikazano na slici 3.2, dobiven je konačni rezultat, gdje su riječi u tekstu jasno označene okvirom, korištenjem primjera logotipa FERIT-a.


*Slika 3.2. Konačan rezultat jednostavnog primjera*

# 4. KOMPLEKSNIJI PRIMJERI IZ MATLAB-A

U ovom poglavlju obrađuje se nekoliko ključnih primjera vezanih uz primjenu dubokog učenja u obradi i klasifikaciji slika. Prvi dio fokusira se na osnovnu klasifikaciju slike, gdje se koristi prilagođena neuronska mreža za prepoznavanje objekata na slikama [5]. Zatim, u sljedećem primjeru, prikazuje se klasifikacija slika korištenjem unaprijed obučenog modela GoogLeNet, koji je sposoban prepoznati veliki broj kategorija objekata uz visoku preciznost [6]. Posljednji primjer demonstrira kako se duboko učenje može primijeniti za detekciju vozila u slikama pomoću specijaliziranog modela za detekciju objekata [7]. Kroz ove primjere prikazat će se postupak od učitavanja podataka i treniranja modela, do evaluacije i primjene na stvarnim slikama.

## 4.1. Klasifikacija slike

### 4.1.1. Učitavanje podataka i podjela u skupove

Podaci uzorka znamenki učitavaju se kao skup slika. Za pristup tim podacima otvara se primjer u obliku „*live script*“ datoteke (Slika 4.1.). Funkcija „*imageDatastore*“ automatski označava slike na temelju naziva mapa. Skup podataka sadrži 10 klasa, a svaka slika unutar skupa ima dimenzije 28x28 piksela s jednom kanalnom vrijednošću.

```
unzip("DigitsData.zip")

imds = imageDatastore("DigitsData", ...,

    IncludeSubfolders=true, ...

    LabelSource="foldernames");

classNames = categories(imds.Labels);
```

*Slika 4.1. MATLAB implementacija skupa podataka*

Podaci se dijele, prema slici 4.2. u skupove za trening, validaciju i testiranje, pri čemu se 70% slika koristi za trening, 15% za validaciju, a preostalih 15% za testiranje. Koristi se nasumična raspodjela ("*randomized*") kako bi se osiguralo da određeni udio datoteka iz svake klase bude ravnomjerno raspoređen u nove skupove podataka. Funkcija „*splitEachLabel*“ koristi se za podjelu skupa slika u tri nova skupa podataka, omogućavajući učinkovitu organizaciju podataka u svrhu treniranja, validacije i testiranja modela.

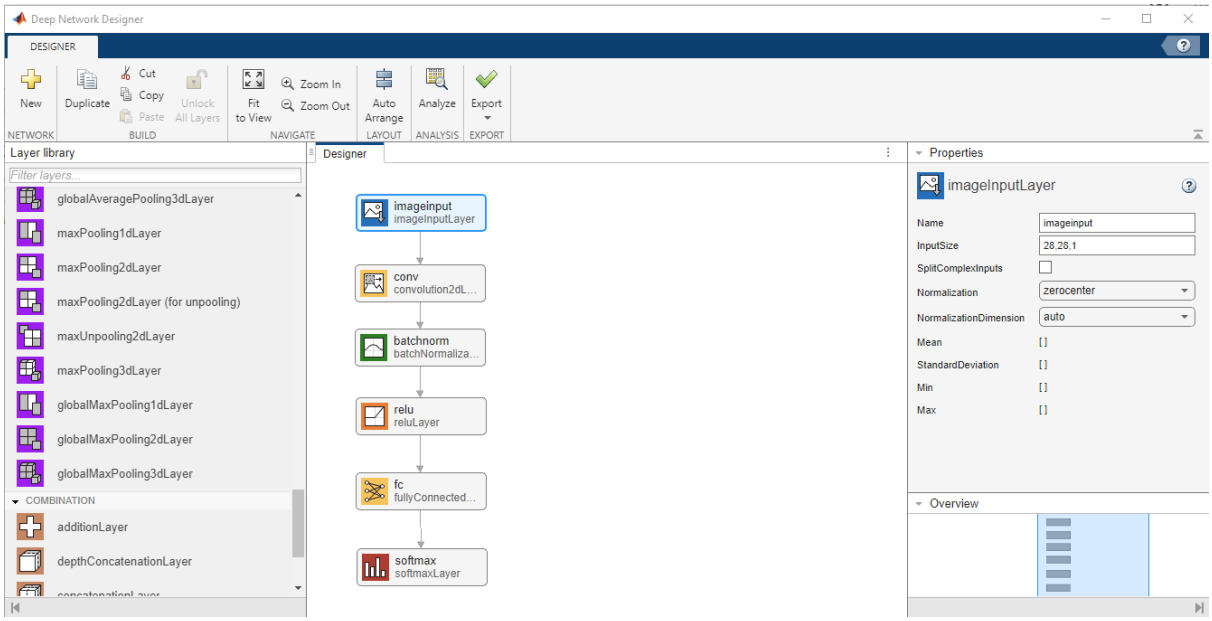
```
[imdsTrain,imdsValidation,imdsTest] = splitEachLabel(imds,0.7,0.15,0.15,"randomized");
```

*Slika 4.2. MATLAB implementacija podjele podataka za trening, validaciju i testiranje*

### 4.1.2. Izgradnja neuronske mreže pomoću alata Deep Network Designer

Za izgradnju neuronske mreže koristi se alat Deep Network Designer. Ova aplikacija omogućava vizualno kreiranje i konfiguriranje složenih arhitektura mreža, olakšavajući tako proces dizajniranja i prilagodbe modela za specifične zadatke.

Aplikacija se koristi tako da se povuku slojevi koji su potrebni za tu neuronsku mrežu, za ovaj primjer se koriste slojevi prema slici. Samo za „*imageInputLayer*“ su promjenjeni parametri za „*InputSize*“ kako bi se dobila veličina slike koja je učitana na početku (Slika 4.3.).




*Slika 4.3. Deep Network Designer u MATLAB-u*

Za provjeru ispravnosti mreže koristi se opcija „*Analyze*“. Ova funkcionalnost omogućava detekciju pogrešaka i osiguranje pravilnog funkcioniranja neuronske mreže. Kada se potvrdi da su

svi elementi mreže ispravno postavljeni, opcija „*Export*“ omogućava spremanje konfigurirane neuronske mreže u MATLAB okruženje.

### 4.1.3. Treniranje mreže podacima

Prema slici 4.4. navode se mogućnosti treninga. Odabir između opcija zahtjeva empirijsku analizu. Istraživanjem različitih konfiguracija opcija treninga se vrši izvođenjem eksperimenata, može se koristiti Experiment Menager app.

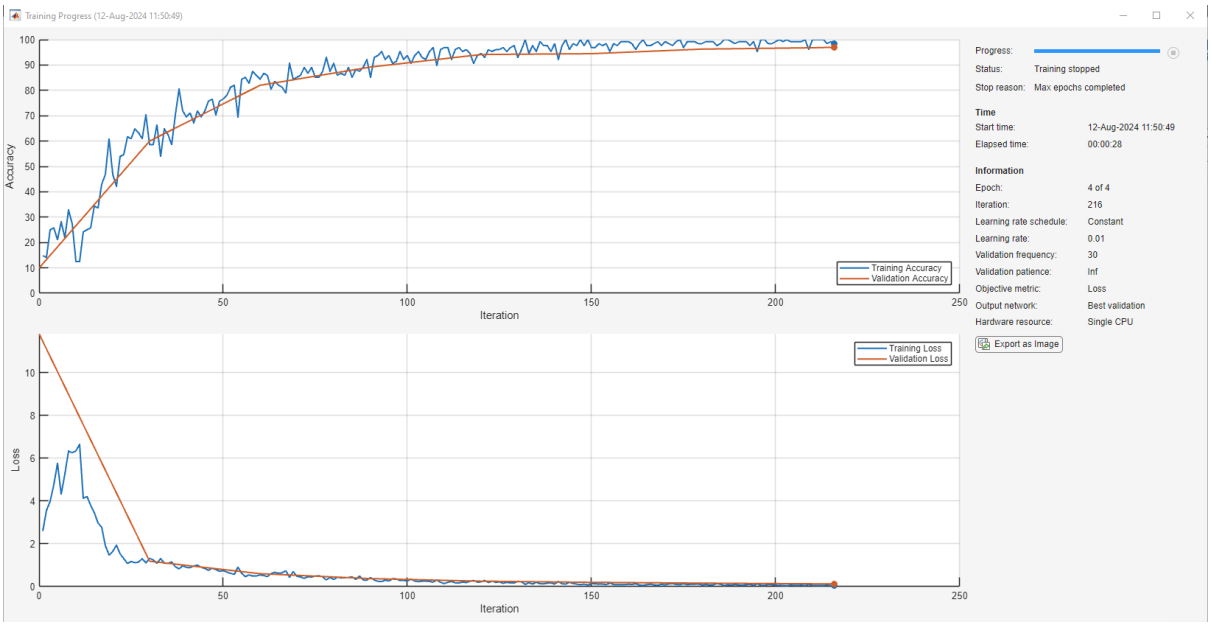
```
options = trainingOptions("sgdm", ...
    MaxEpochs=4, ...
    ValidationData=imdsValidation, ...
    ValidationFrequency=30, ...
    Plots="training-progress", ...
    Metrics="accuracy", ...
    Verbose=false);
```

*Slika 4.4. MATLAB implementiranje mogućnosti treninga*

Treba uvježbati neuronsku mrežu pomoću funckije „*trainnet*“. Budući da je cilj klasifikacija, upotrebljava se gubitak unakrsne entropije kao što je prikazano na slici 4.5.

```
net = trainnet(imdsTrain,net_1,"crossentropy",options);
```

*Slika 4.5. MATLAB implementacija unakrsne entropije*




*Slika 4.6. Napredak treninga unakrsne entropije*

### 4.1.4. Klasifikacija validacijskih podataka i vizualizacija

Za testiranje neuronske mreže, prema slici 4.7. klasificiraju se validacijski podaci i izračunava se točnost klasifikacije.

```
scores = minibatchpredict(net,imdsValidation);

YValidation = scores2label(scores,classNames);

TValidation = imdsValidation.Labels;

accuracy = mean(YValidation == TValidation);
```

*Slika 4.7. MATLAB klasifikacija validacijskih podataka i izračun točnosti klasifikacije*

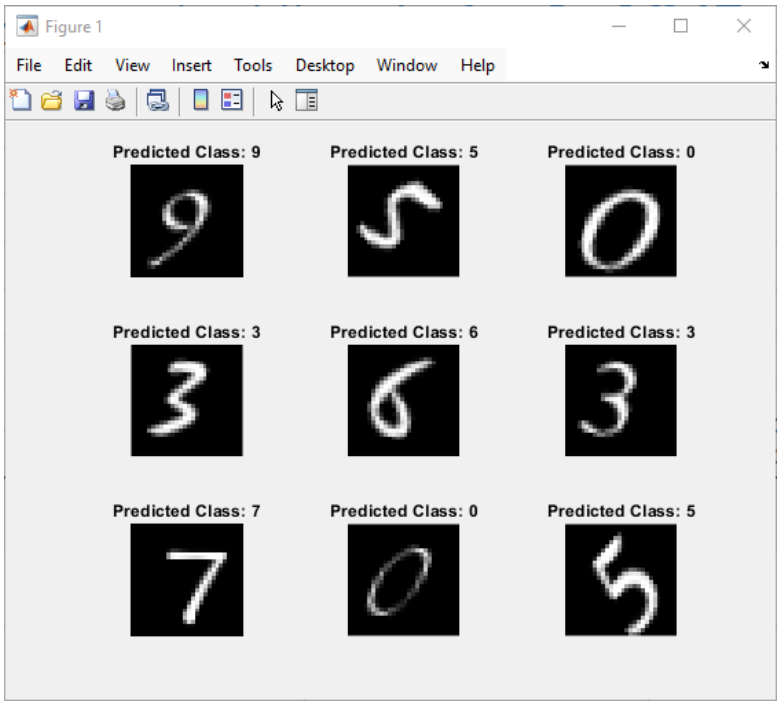
Nakon toga, točnost modela je izračunata i iznosi 0.9780.

Dalje, vizualizirane su odabrane predikcije kako bi se prikazala učinkovitost modela u prepoznavanju i klasifikaciji uzoraka.

```
numValidationObservations = numel(imdsValidation.Files);

idx = randi(numValidationObservations,9,1);
figure

tiledlayout("flow")

for i = 1:9

    nexttile

    img = readimage(imdsValidation,idx(i));

    imshow(img)

    title("Predicted Class: " + string(YValidation(idx(i))))

end
```

*Slika 4.8. MATLAB implementacija vizualizacije predikcije*

Nakon implementacije predikcije dobiva se vizualizacija podataka, prema slici 4.9.




*Slika 4.9. Vizualizacija predikcije*

## 4.2. Klasifikacija Slike Korištenjem Modela GoogLeNet

Ovaj primjer ilustrira postupak klasifikacije slika korištenjem prethodno trenirane duboke konvolucijske neuronske mreže GoogLeNet.

GoogLeNet je model obučen na više od milijun slika i sposoban je klasificirati slike u 1000 različitih kategorija objekata, poput tipkovnica, šalica za kavu, olovki, te različitih životinja.

### 4.2.1. Učitavanje unaprijed izvježbane mreže

Mreža je razvila bogate prikaze značajki, što joj omogućuje prepoznavanje širokog spektra vizualnih obrazaca. Prilikom obrade slike, GoogLeNet prima sliku kao ulazni podatak te generira oznaku za prepoznati objekt na slici, zajedno s vjerojatnostima za svaku od mogućih kategorija objekata.

Za učitavanje unaprijed obučene GoogLeNet mreže i odgovarajućih naziva klasa, koristi se funkcija „*imagePretrainedNetwork*“. Ovaj korak zahtijeva instaliran Deep Learning Toolbox™ Model paket podrške za GoogLeNet mrežu. Ukoliko potrebni paketi podrške nisu instalirani, softver će ponuditi vezu za njihovo preuzimanje i instalaciju.

Slika koju je potrebno klasificirati mora imati istu veličinu kao i ulazna veličina mreže. Kod GoogLeNet-a, prvi element svojstva „*Layers*“ mreže predstavlja ulazni sloj za slike. Veličina ulazne slike definirana je svojstvom „*InputSize*“ tog ulaznog sloja.

Prema slici 4.10. ovaj kod prikazuje nasumično odabranih 10 naziva klasa iz ukupnog skupa svih klasa. Varijabla „*numClasses*“ definira ukupan broj dostupnih klasa, dok funkcija „*randperm*“ generira nasumični poredak tih klasa. Na temelju ovog poretka, zatim se odabire i prikazuje 10 nasumičnih klasa.

```
[net,classNames] = imagePretrainedNetwork("googlenet");

inputSize = net.Layers(1).InputSize;

numClasses = numel(classNames);

disp(classNames(randperm(numClasses,10)))
```

*Slika 4.10. MATLAB učitavanje unaprijed izvježbane mreže*

Učitavanjem slike prema prikazu na slici 4.11, dobivamo sliku prikazanu na slici 4.12.

```
I = imread("peppers.png");

figure

imshow(I)
```

*Slika 4.11. MATLAB učitavanje slike*

*Slika 4.12. Učitana slika*

### 4.2.2. Promjena veličine i klasifikacija slike

Prikaz dimenzija slike: Slika ima veličinu od 384 puta 512 piksela i sastoji se od tri kanala boja (RGB). Kako bi se slika prilagodila ulaznoj veličini mreže, potrebno ju je skalirati pomoću funkcije „*imresize*“. Ova promjena veličine može neznatno izmijeniti omjer stranica slike (Slika 4.13.).

```
size(I)

X = imresize(I,inputSize(1:2));

figure

imshow(X)
```

*Slika 4.13. MATLAB promjena veličine slike*

Prema slici 4.14, za izvođenje predviđanja koristeći neuronsku mrežu, koristi se funkcija predviđanja na jednu sliku. Ako slika ima tip podataka „*uint8*“, prvo se treba pretvoriti u tip podataka „*single*“ kako bi se omogućila pravilna obrada mrežom. Za korištenje GPU-a, podatke treba pretvoriti u „*gpuArray*“.

```
X = single(X);
if canUseGPU
    X = gpuArray(X);
end
scores = predict(net,X);
```

*Slika 4.14. MATLAB predviđanje slike*

Funkcija za predviđanje vraća vjerojatnosti za svaku klasu. Za pretvaranje rezultata klasifikacije u odgovarajuće kategoričke oznake, koristi se funkcija „*scores2label*“ (Slika 4.15.). Nakon toga, izvorni prikaz slike prikazuje se zajedno s predviđenom oznakom i pripadajućom vjerojatnošću, koja pokazuje koliko je velika šansa da slika pripada toj oznaci, kao što je prikazano na slici 4.16.

```
[label,score] = scores2label(scores,classNames);
figure
imshow(I)
title(string(label) + ", " + score)
```

*Slika 4.15. MATLAB prikaz originalne slike sa predikcijom*

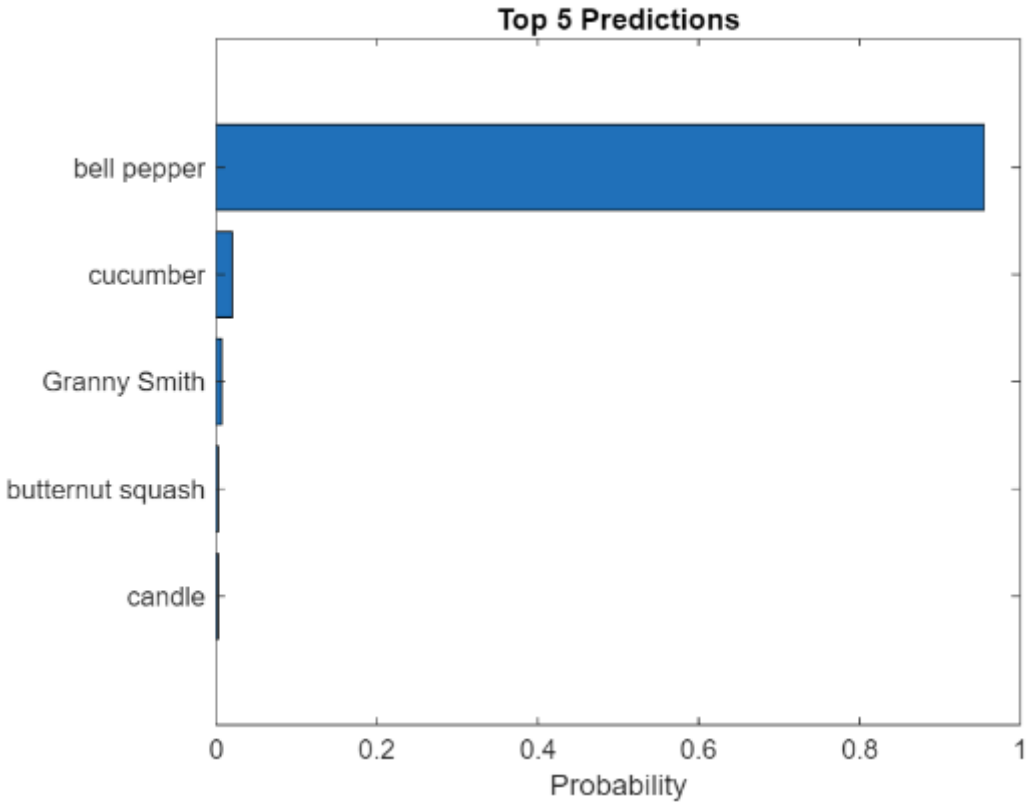

*Slika 4.16. Prikaz originalne slike sa predikcijom*

### 4.2.3. Prikaz najboljih predviđanja

Prema slici 4.17. prikazuju se prvih pet predviđenih oznaka zajedno s pripadajućim vjerojatnostima u obliku histograma. S obzirom na to da mreža može klasificirati slike u veliki broj kategorija, od kojih su mnoge slične, uobičajeno je razmotriti točnost prvih pet predviđanja prilikom evaluacije performansi mreže. U ovom slučaju, mreža s visokom vjerojatnošću klasificira sliku kao papriku (Slika 4.18.).

```
[~,idx] = sort(scores,"descend");

idx = idx(5:-1:1);

classNamesTop = classNames(idx);

scoresTop = scores(idx);

figure

barh(scoresTop)

xlim([0 1])

title("Top 5 Predictions")

xlabel("Probability")

yticklabels(classNamesTop)
```

*Slika 4.17. MATLAB prikaz najboljih predviđanja*




*Slika 4.18. Prikaz najboljih predviđanja*

## 4.3. Detektor vozila za duboko učenje

Detekcija vozila pomoću računalnog vida ključna je za praćenje vozila oko ego vozila. Sposobnost otkrivanja i praćenja vozila bitna je za mnoge aplikacije autonomne vožnje, uključujući sustave upozorenja na sudar sprijeda, prilagodljive tempomate i automatsko održavanje vozne trake. Automated Driving Toolbox™ nudi prethodno obučene detektore vozila, kao što su vehicleDetectorFasterRCNN i vehicleDetectorACF, koji omogućuju brzu izradu prototipova. Međutim, unaprijed obučeni modeli možda neće odgovarati svim specifičnim aplikacijama, zbog čega je ponekad potrebno trenirati modele od početka. Ovaj primjer demonstrira kako trenirati detektor vozila od nule koristeći tehnike dubokog učenja.

### 4.3.1. Preuzimanje unaprijed izvježbanog detektora

Preuzimanje prethodno izvježbanog detektora omogućuje izbjegavanje dugotrajnog čekanja na završetak obuke. Ukoliko je potrebno obučiti detektor od početka, varijabla „*doTraining*“ treba biti postavljena na „*true*“ (Slika 4.19).

```
doTraining = false;

if ~doTraining && ~exist("fasterRCNNResNet50EndToEndVehicleExample.mat" ...
        ,"file")

    disp("Downloading pretrained detector (118 MB)...");

    pretrainedURL = "https://www.mathworks.com/supportfiles/vision/" + ...
        "data/fasterRCNNResNet50EndToEndVehicleExample.mat";

    websave("fasterRCNNResNet50EndToEndVehicleExample.mat",pretrainedURL);

end
```

*Slika 4.19. MATLAB preuzimanje izvježbanog detektora*

### 4.3.2. Učitavanje skupa podataka

Ovaj primjer (Slika 4.20.) koristi mali označeni skup podataka koji se sastoji od 295 slika. Većina slika potječe iz skupova podataka Caltech Cars 1999. i 2001., dostupnih na web stranici Caltech Computational Vision, koju je razvio Pietro Perona i koristi se uz dopuštenje. Svaka slika sadrži jednu ili dvije označene instance vozila. Iako je ovaj mali skup podataka koristan za istraživanje procesa obuke bržeg R-CNN-a, u praksi je potrebno više označenih slika za obuku robusnijeg detektora. Preporučuje se raspakiravanje slika vozila i učitavanje pravih podataka o vozilu za detaljniju obuku.

```
unzip vehicleDatasetImages.zip

data = load("vehicleDatasetGroundTruth.mat");

vehicleDataset = data.vehicleDataset;
```

*Slika 4.20. MATLAB učitavanje skupa podataka*

Podaci o vozilu pohranjeni su u tablici s dva stupca (Slika 4.21.): prvi stupac sadrži putanje slikovnih datoteka, dok drugi stupac uključuje granične okvire vozila.

Skup podataka treba podijeliti na dva dijela: jedan za trening detektora i drugi za testiranje njegovih performansi. Preporučuje se da se 60% podataka koristi za obuku, dok preostalih 40% bude rezervirano za evaluaciju detektora.

```
rng(0)

shuffledIdx = randperm(height(vehicleDataset));

idx = floor(0.6 * height(vehicleDataset));

trainingDataTbl = vehicleDataset(shuffledIdx(1:idx),:);

testDataTbl = vehicleDataset(shuffledIdx(idx+1:end),:);
```

*Slika 4.21. MATLAB podjela podataka za trening i testiranje*

Za kreiranje spremišta podataka koje omogućava učitavanje slika i pripadajućih oznaka tijekom obuke i evaluacije, koristi se „*imageDatastore*“ i „*boxLabelDatastore*“ (Slika 4.22.). Ovi alati omogućuju učinkovito upravljanje i pristup podacima u fazama obuke i testiranja modela.

```
imdsTrain = imageDatastore(trainingDataTbl{:,"imageFilename"});

bldsTrain = boxLabelDatastore(trainingDataTbl(:,"vehicle"));

imdsTest = imageDatastore(testDataTbl{:,"imageFilename"});

bldsTest = boxLabelDatastore(testDataTbl(:,"vehicle"));
```

*Slika 4.22. MATLAB kreiranje spremišta*

Kombiniraju se spremišta podataka slika i oznaka graničnih okvira prema slici 4.23., kako bi se omogućio integrirani pristup svim relevantnim informacijama potrebnim za obuku i evaluaciju modela.

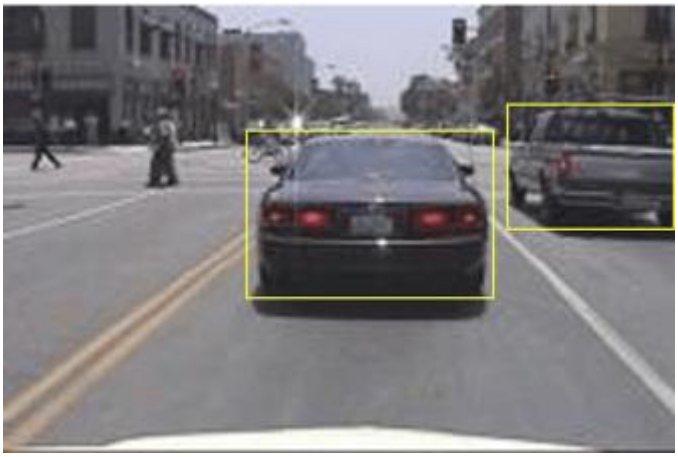
```
trainingData = combine(imdsTrain,bldsTrain);

testData = combine(imdsTest,bldsTest);
```

*Slika 4.23. MATLAB kombiniranje spremišta*

Prikazuje se jedna od slika za trening zajedno s pripadajućim oznakama graničnih okvira. Kod prikazan na slici 4.24. sa rezultantnom slikom 4.25.

```
data = read(trainingData);

I = data{1};

bbox = data{2};

annotatedImage = insertShape(I,"rectangle",bbox);

annotatedImage = imresize(annotatedImage,2);

figure

imshow(annotatedImage)
```

*Slika 4.24. MATLAB prikaz jedne slike za trening s pripadajućim oznakama*



*Slika 4.25. Prikaz jedne slike za trening s pripadajućim oznakama*

# 5. IZRADA LABARATORIJSKE VJEŽBE

## 5.1. Detekcija teksta na slici

U ovom uvodnom zadatku cilj je steći osnovno razumijevanje neuronskih mreža kroz jednostavan primjer u MATLAB-u. Zadatak je osmišljen tako da ilustrira proces prepoznavanja riječi na slici, pri čemu neuronska mreža identificira prisutne riječi i označava ih odgovarajućim okvirima. Na taj način, zadatak služi kao praktičan uvod u primjenu neuronskih mreža za analizu i obradu slika, omogućujući korisnicima da se upoznaju s osnovnim tehnikama i metodologijama korištenja u ovom području.

Prije početka rada na zadatku, potrebno je proučiti model dubokog učenja za prepoznavanje regije znakova poznat kao CRAFT (Character Region Awareness for Text detection). Funkcija koja koristi ovaj model za detekciju teksta naziva se „*detectTextCraft()*“.

### 5.1.1. Zadatak za samostalni rad

Nakon proučavanja modela, prvi korak je učitavanje slike koja sadrži tekst. Zatim je potrebno koristiti funkciju „*insertShape*“ za crtanje graničnih okvira oko prepoznatog teksta na slici. Na kraju, treba prikazati tako modificiranu sliku s dodanim graničnim okvirima. Očekivani rezultat trebao bi biti sličan jednostavnom primjeru prikazanom u trećem poglavlju.

## 5.2. Klasifikacija slike

U ovom zadatku cilj je obučiti model koristeći Deep Network Designer kako bi mogao precizno predvidjeti kojoj klasi pripada svaka slika. Konkretno, u ovom primjeru koristi se direktorij sa slikama rukom pisanih brojeva, pri čemu je zadatak modela predvidjeti koji broj predstavlja svaka slika.

### 5.2.1. Zadatak za samostalni rad

Skup podataka se učitava kao na slici 5.1. Skup podataka ima 10 klasa i svaka slika u skupu podataka je 28x28 po 1 piksel. Funkcija imageDatastore automatski označava slike na temelju naziva mapa.

Skup podataka učitava se prema postupku prikazanom na slici 5.1. Ovaj skup podataka sadrži 10 različitih klasa, pri čemu svaka slika ima dimenzije 28x28 piksela s jednom kanalnom razinom.

Funkcija „*imageDatastore*“ automatski označava slike prema nazivima mapa, olakšavajući organizaciju i pristup podacima tijekom obrade.

```
unzip("DigitsData.zip")

imds = imageDatastore("DigitsData", ...,

    IncludeSubfolders=true, ...

    LabelSource="foldernames");

classNames = categories(imds.Labels);
```

*Slika 5.1. MATLAB implementacija skupa podataka*

Podatke je potrebno podijeliti u skupove za obuku, validaciju i testiranje. Preporučuje se korištenje 70% podataka za obuku, 15% za validaciju i 15% za testiranje. Ova podjela se provodi pomoću funkcije „*splitEachLabel*“, koja koristi prethodno definiranu varijablu „*imds*“. Kako bi se osigurala nasumična raspodjela podataka, funkciji je potrebno dodati argument „*"randomized"*“.

Za izgradnju mreže koristi se aplikacija Deep Network Designer, koja se pokreće funkcijom „*deepNetworkDesigner*“. Prilikom kreiranja mreže, potrebno je dodati i međusobno povezati slojeve sljedećim redoslijedom: imageInputLayer, convolution2dLayer, batchNormalizationLayer, reluLayer, fullyConnectedLayer, te na kraju softmaxLayer. Od svih slojeva, jedino je potrebno prilagoditi sloj imageInputLayer kako bi veličina ulaznih slika odgovarala zadanom „*InputSize*“.

Kako bi se provjerila ispravnost mreže koristi se tipka Analyze koja služi za provjeru pogrešaka i provjeru ispravnosti neuronske mreže. Kada je sve kako treba, koristi se tipka Export koja sprema neuronsku mrežu u MATLAB.

U skladu s mogućnostima prikazanim na slici 5.2., postavke obuke treba pažljivo konfigurirati. Nakon toga, potrebno je obučiti neuronsku mrežu koristeći funkciju „*trainnet*“.

```
options = trainingOptions("sgdm", ...
    MaxEpochs=4, ...
    ValidationData=imdsValidation, ...
    ValidationFrequency=30, ...
    Plots="training-progress", ...
    Metrics="accuracy", ...
    Verbose=false);
```

*Slika 5.2. MATLAB definiranje trening opcija*

Za testiranje neuronske mreže, validacijski podaci se klasificiraju, a zatim se izračunava točnost klasifikacije, kako je prikazano na slici 5.3.

```
scores = minibatchpredict(net,imdsValidation);
YValidation = scores2label(scores,classNames);
TValidation = imdsValidation.Labels;
accuracy = mean(YValidation == TValidation);
```

*Slika 5.2. MATLAB validacijski podaci i računanje točnosti*

Potrebno je zabilježiti postignutu točnost modela. Nadalje, potrebno je vizualizirati odabrane predikcije, kao što je prikazano na slici 5.3. Rezultat ove vizualizacije uključuje prikaz devet slika s pripadajućim predviđenim klasama koje su navedene iznad svake slike.

```
numValidationObservations = numel(imdsValidation.Files);

idx = randi(numValidationObservations,9,1);
figure

tiledlayout("flow")

for i = 1:9

    nexttile

    img = readimage(imdsValidation,idx(i));

    imshow(img)

    title("Predicted Class: " + string(YValidation(idx(i))))

end
```

*Slika 5.3. MATLAB vizualizacija predikcije*

# 6. ZAKLJUČAK

U ovom završnom radu ostvareni su ključni ciljevi postavljeni na početku istraživanja, a koji su se odnosili na primjenu neuronskih mreža unutar okruženja MATLAB-a. Kroz rad je prikazana implementacija neuronske mreže, počevši od pripreme i podjele podataka na skupove za obuku, validaciju i testiranje, preko izgradnje mreže korištenjem alata Deep Network Designer, do obuke modela i evaluacije rezultata.

Jedan od najznačajnijih rezultata ovog istraživanja je uspješna implementacija neuronske mreže koja je ostvarila visoku točnost klasifikacije, što potvrđuje učinkovitost pristupa odabranog za ovu analizu. Posebno su detaljno obrađeni koraci u pripremi podataka i dizajnu mreže, čime su prikazane prednosti MATLAB-a kao alata za duboko učenje, uključujući intuitivnost korisničkog sučelja i široke mogućnosti prilagodbe modela. Također, vizualizacija rezultata predikcija dodatno je doprinijela razumijevanju rada mreže i omogućila jasniji prikaz postignutih rezultata.

Međutim, ograničenja ovog rada leže u korištenju relativno malog skupa podataka, što može utjecati na generalizaciju modela u stvarnim aplikacijama. Nadalje, iako su rezultati zadovoljavajući, postoji prostor za unaprjeđenje, posebice u optimizaciji mreže i proširenju skupa podataka, što bi moglo rezultirati još većom točnošću klasifikacije.

Kao smjernice za daljnji rad, predlaže se istraživanje naprednijih arhitektura neuronskih mreža, kao i primjena transfernog učenja kako bi se iskoristili već obučeni modeli na većim skupovima podataka. Također, dodatna optimizacija hiperparametara i proširenje skupa podataka moglo bi doprinijeti poboljšanju rezultata, čime bi se proširile mogućnosti primjene ovog modela u stvarnim problemima.

# LITERATURA

[1] „MathWorks. "Deep Learning Toolbox." MATLAB Documentation“.
https://www.mathworks.com/products/deep-learning.html [26.6.2024.]
[2] „TensorFlow, "TensorFlow Tutorials," TensorFlow, 2024.“:
https://www.tensorflow.org/tutorials [24.8.2024.]
[3] „PyTorch, "Intro to PyTorch," PyTorch Tutorials, 2024.“:
https://pytorch.org/tutorials/beginner/basics/intro.html [24.8.2024.]
[4] „MathWorks. "Deep Learning Toolbox." MATLAB Documentation“.
https://www.mathworks.com/help/vision/ug/recognize-text-using-craft-model-and-ocr.html
[24.8.2024.]
[5] „MathWorks. "Deep Learning Toolbox." MATLAB Documentation“.
https://www.mathworks.com/help/deeplearning/gs/create-simple-image-classification-network-using-deep-network-designer.html [26.8.2024.]
[6] „MathWorks. "Deep Learning Toolbox." MATLAB Documentation“.
https://www.mathworks.com/help/deeplearning/ug/classify-image-using-googlenet.html
[26.8.2024.]
[7] „MathWorks. "Deep Learning Toolbox." MATLAB Documentation“.
https://www.mathworks.com/help/deeplearning/ug/train-a-deep-learning-vehicle-detector.html
[27.8.2024.]

# SAŽETAK

U završnom radu istražena je primjena neuronskih mreža u MATLAB-u kroz laboratorijsku vježbu usmjerenu na jedan konkretan primjer. Glavni problem rada bio je razvoj i evaluacija neuronske mreže za klasifikaciju slika, s ciljem postizanja visoke točnosti u prepoznavanju predmeta na slikama. Za rješavanje ovog problema korišten je alat Deep Network Designer unutar MATLAB-a za izgradnju i obuku neuronske mreže. Proces je obuhvaćao pripremu skupa podataka, konstrukciju modela, trening mreže i evaluaciju rezultata. Kroz laboratorijsku vježbu, implementiran je jednostavan primjer klasifikacije slika rukom pisanih brojeva, gdje su se prikazali svi ključni koraci od učitavanja podataka do evaluacije performansi modela. Postignuti rezultati uključuju visoku točnost klasifikacije modela, što potvrđuje učinkovitost pristupa primijenjenog u radu. Ovaj primjer pokazuje praktične aspekte primjene neuronskih mreža u MATLAB-u i pruža osnovu za daljnje istraživanje i primjenu u stvarnim scenarijima.

**Ključne riječi:** Deep Network Designer, klasifikacija slike, MATLAB, neuronske mreže, trening modela

# ABSTRACT

**Modeling and simulation of neural networks using MATLAB**

This final paper explores the application of neural networks in MATLAB through a laboratory exercise focused on a specific example. The main problem addressed was the development and evaluation of a neural network for image classification, with the goal of achieving high accuracy in object recognition within images. To tackle this problem, the Deep Network Designer tool within MATLAB was used to build and train the neural network. The process included data preparation, model construction, network training, and result evaluation. The laboratory exercise implemented a simple example of classifying handwritten digits, demonstrating all key steps from data loading to performance evaluation of the model. The achieved results include high classification accuracy, confirming the effectiveness of the approach applied in the work. This example illustrates the practical aspects of using neural networks in MATLAB and provides a foundation for further research and application in real-world scenarios.



**Key words:** Deep Network Designer, image classification, MATLAB, model training, neural networks

# ŽIVOTOPIS

Lovro Ružman rođen je 3. veljače 2002. godine u Osijeku u Hrvatskoj. Pohađao je Osnovnu školu Matije Petra Katančića u Valpovu. Nakon završene osnovne škole upisuje srednju školu za Elektrotehničara u Valpovu. Maturirao je 2020. godine. Iste godine upisuje preddiplomski studij Elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Nakon završene prve godine Elektrotehnike se prebacuje na preddiplomski studij Računarstva na istom fakultetu. Nakon neuspjele druge godine u ponavljačkoj godini upisuje i završava mikro kvalifikaciju izrada web stranica obujma 7 CSVET bodova u trajanju od 175 sati na Visokom učilištu Algebra.

_____
Lovro Ružman

# PRILOZI

GITHUB

- Svi kodovi i primjeri se nalaze na linku: https://github.com/lovroruzman/ZAVRSNI_RAD