SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU ELEKTROTEHNIČKI FAKULTET

Diplomski studij računarstva

Laboratorijska vježba 6

Raspoznavanje znamenaka korištenjem konvolucijskih neuronskih mreža

Ivan Budoš, DRB

Osijek, 2023.

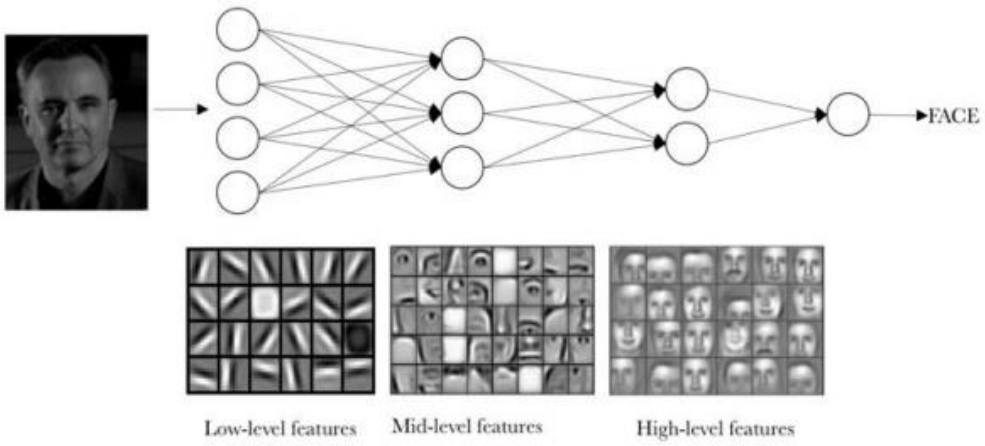
SADRŽAJ

UVOD

U šestoj laboratorijskoj vježbi potrebno je projektirati i ispitati neuronsku mrežu koja služi za raspoznavanje znamenaka. Korištena baza za treniranje i učenje je MNIST baza podataka koja sadrži 70000 slika znamenaka od kojih se 60000 koristi za treniranje a 10000 za testiranje.. Potrebno je izraditi standardnu neuronsku mrežu koju je također potrebno naučiti da raspoznaje znamenke korištenjem iste baze podataka. Cilj vježbe je usporediti standardnu i konvolucijsku neuronsku mrežu i pobliže opisati dobivene rezultate usporedbe.

KONVOLUCIJSKE NEURONSKE MREŽE

Duboko učenje je grana strojnog učenja, dok je konvolucijska neuronska mreža konkretna implementacija dubokih neuronskih mreža koja se najčešće primjenjuje za analizu i raspoznavanje slika. Standardne neuronske mreže nisu osobito prigodne u svrhu analize slika zbog zahtjeva da svaki neuron mora biti povezan sa svim elementima iz prethodnog sloja. Stoga ako npr. imamo sliku veličine 25 x 25 što znaci ukupno 225 piksela u konačnici će rezultirati s 225 težina koje je potrebno podesiti i optimizirati samo za jedan jedini neuron. Stoga, ako je je slika skromnih 100 x 100 piksela to je već 10000 težina po neuronu i ako imamo 100 neurona u jednom sloju ispada 1000000 težina koje se moraju podesiti. Konvolucijske neuronske mreže imaju takvu arhitekturu koja omogućava da se raspoznaju uzorci različitih dimenzija uz puno manje parametara koje je potrebno podesiti i optimizirati. Smisao konvolucijske neuronske mreže jest da se kroz slojeve neuronske mreže uče raspoznavati inkrementalno kompleksniji uzorci. Ako se uči npr. raspoznavanje ljudskih lica, na prvoj razini se raspoznaju linije, rubovi i različiti točkasti uzorci, na drugoj su to npr. dijelovi lica kao što je to nos, uho, usta, oko i sl., dok na idućoj razini su to veći dijelovi lica itd.

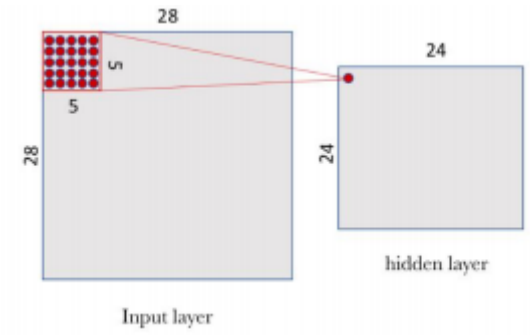


Slika 1. Kompleksnost raspoznavanja uzoraka se inkrementalno povećava kroz slojeve

Drugim riječima, svaki sloj konvolucijske neuronske mreže uči različitu razinu apstrakcije ulaznih podataka. Iako konvolucijska neuronska mreža može imati mnogo različitih dijelova, tipični dijelovi uključuje sljedeće slojeve:

- Konvolucijski sloj (engl. Convolution layer)
- Sloj sažimanja (engl. Pooling Layer),
- Potpuno povezani sloj (engl. Fully-Connected (dense) Layer)

Unutarnja struktura tipične konvolucijske neuronske mreže sastoji se od nekoliko naizmjenično poslaganih višedimenzionalnih konvolucijskih slojeva i slojeva sažimanja. Na kraju se nalazi potpuno povezan sloj, koji je jednodimenzionalan, kao i izlazni sloj. Fundamentalna razlika između konvolucijskog sloja i potpuno povezanog sloja (istovrstan standardnoj neuronskoj mreži), jest to što potpuno povezan sloj uči uzorke globalno dok konvolucijski sloj uči lokalne uzorke unutar malih dvodimenzionalnih prozora nazvanih jezgra (engl. kernel). Tako naučne lokalne uzorke konvolucijski sloj nauči raspoznavati na bilo kojem mjestu na slici, dok bi na primjer standardne neuronske mreže naučile raspoznavati samo na točno određenom mjestu. Stoga ako bi neka značajka promijenila svoj položaj, standardna neuronska mreža bi morala ponovo biti naučena. Još jedna bitna značajka konvolucijskih slojeva je ta da može naučiti odnosno zadržati prostorne odnose između uzoraka. Ako se na primjer u prvom konvolucijskom sloju uče osnovni uzorci kao što su rubovi, linije, prijelazi i sl. tada se u drugom sloju mogu naučiti kompleksne kompozicije tih osnovnih uzoraka iz prethodnog sloja. Time se iz sloja u sloj povećava kompleksnost apstraktnih vizualnih koncepata. Konvolucijski slojevi i slojevi sažimanja se baziraju na 3D tenzorima nazvanima mape značajki (engl. feature map), koje imaju širinu, visinu i dubinu. Ulazni slojevi mogu isto imati te tri dimenzije, ako se radi o crno-bijeloj slici tada pored širine i visine ima samo dubinu od 1, dok slike u boji imaju dubinu 3 (crvena, zelena i plava komponenta slike). Konvolucijski sloj radi na principu konvolucije gdje maleni prozor „klizi“ po slici, umnaža elemente (piksele) slike i potom ih sumira, te uz dodatak bias-a „pridružuje“ neuronu na odgovarajućem mjestu konvolucijskog sloja. Vizualno, obradu krećemo s prozorom u gornjem lijevom kutu slike koji daje potrebne informacije prvom (opet, gornji lijevi kut) neuronu konvolucijskog sloja. Nakon toga pomičemo prozor za jedno mjesto u desno i povezujemo vrijednosti s drugim neuronom u konvolucijskom sloju i tako obilazimo cijelu sliku, s lijeva na desno, od gore prema dolje.

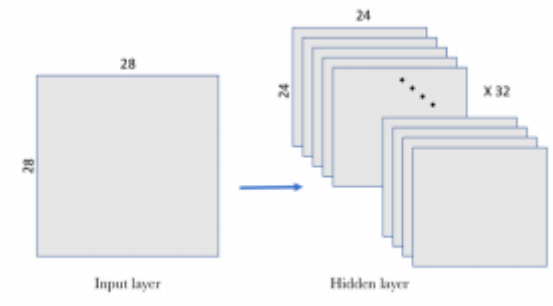


Slika 2. Konvolucija se obavlja prozorima predefinirane veličine

Ako imamo ulaznu sliku 28 x 28 pikesela (npr. MNIST baza slika znamenki) i prozor od 5 x 5, tada veličina konvolucijskog sloja može biti samo 24 x 24, jer nam ta dimenzija omogućava da obiđemo sve elemente ulazne slike s našim prozorom bez da narušavamo granice slike, odnosno možemo pomaknuti prozor samo za 23 mjesta u desno (odnosno prema dolje) bez da pređemo granicu slike. Naravno prozor se po želji može pomicati za više mjesta odjednom što se definira kao korak (engl. stride) uzorkovanja, a također na rubove slike se mogu dodavati odgovarajući broj nula čime se omogućava da unutarnji (konvolucijski) slojevi neuronske mreže imaju istu širinu i visinu kao i ulazna slika (engl. padding).

Dakle prema korištenom primjeru, svaki neuron iz skrivenog sloja je povezan s odgovarajućom regijom ulazne slike preko 5 x 5 prozora. To povezivanje je definirano s 5 x 5 matricom težina koja se naziva filter i pripadajućim bias-om. Potrebno je naglasiti da se ista matrica težina i isti bias koristi za sve neurone određenog podsloja unutar nekog konvolucijskog sloja neuronske mreže. Dakle samo te težine i bias se podešavaju tijekom učenja neuronske mreže (za svaki sloj). Što npr. znači da za jedan sloj gdje se koristi 5 x 5 filter podešava 26 vrijednost (25 težina + 1 bias) za razliku od npr. standardne neuronske mreže kada imamo 28 x 28 sliku na ulazu, podešavalo bi se 784 težine za svaki neuron, a ovdje samo 26 za svaki sloj.

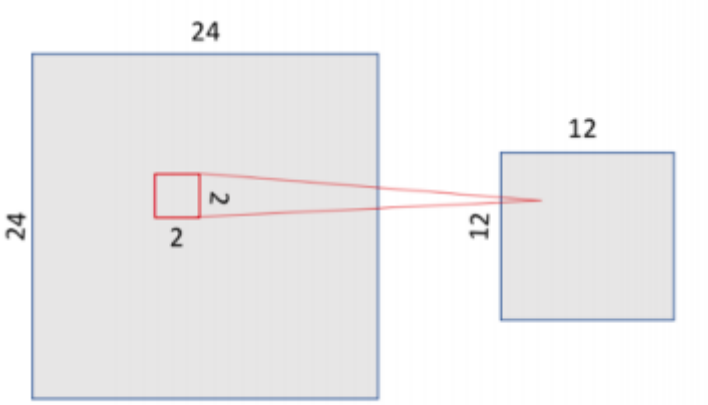
Filteri koji se ovdje koriste se mogu povezati s filterima koji se koriste za standardnu obradu slike (detekcija rubova, izoštravanje, zamućivanje i sl.). Individualni filtri mogu detektirai samo jedan tip značajki, stoga je preporučano da se koristi više filtera u isto vrijeme koji će detektirati različite značajke. U konvolucijskim slojevima se stoga nalazi više podslojeva istih dimenzija, svaki sa svojim filterom tj. težinama i bias-om.



Slika 3. Svaki konvolucijski sloj tipično ima više filtera, svaki povezan sa svojim podslojem

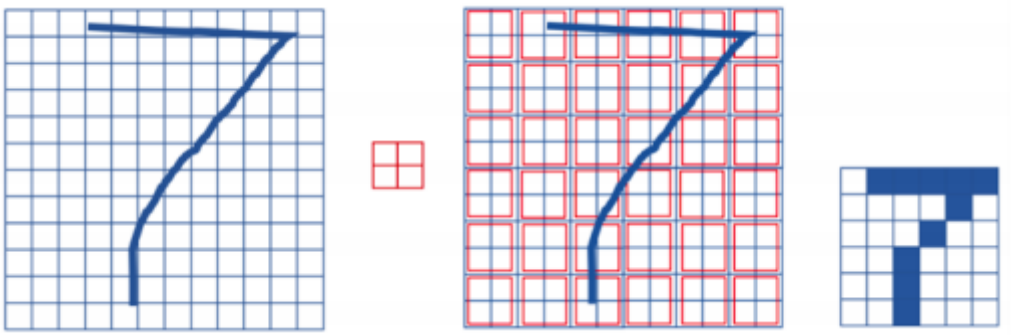
U primjeru prikazanom na Slici 3, prvi konvolucijski sloj prima tenzor dimenzija (28,28,1) i rezultira tenzorom (24,24,32) za koji je potrebno proračunati i optimizirati 32 x 5 x 5 + 32 x 1

= 832 vrijednosti. Često se nakon konvolucijskih slojeva postavljaju slojevi sažimanja (engl. pooling). Njihova je svrha da pojednostave informaciju koja dolazi iz konvolucijskih slojeva i time kondenzira njihov sadržaj. Drugim riječima funkcija sažimanja je da mapira skup prostorno bliskih značajki na ulazu u jednu značajku na izlazu, u tu svrhu se koristi statistički pokazatelji kao što su maksimum ili srednja vrijednost. Ako npr. u našem slučaju koristimo funkciju sažimanja s 2 x 2 prozorom, dobit ćemo na izlazu tenzor s duplo manjom širinom i visinom



Slika 4. Sloj sažimanja rezultira s manjom količinom podataka koje je potrebno obraditi u narednim slojevima

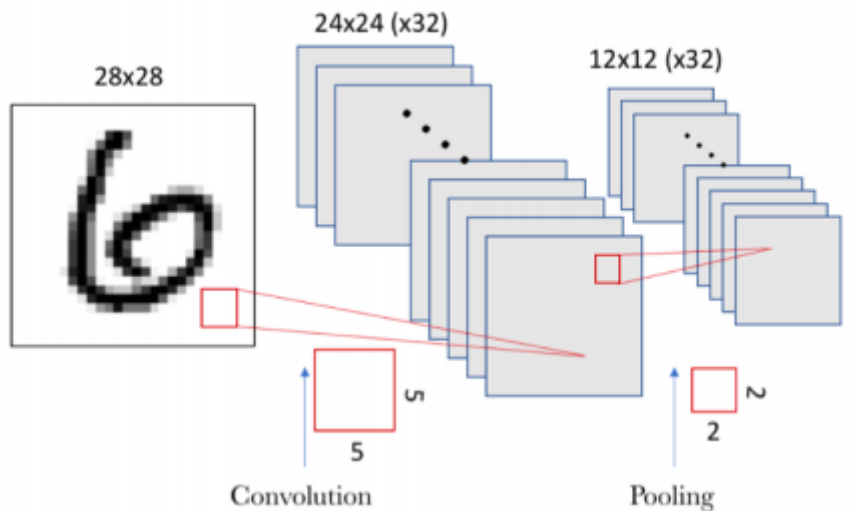
Ovakvim pristupom sažimanja se i dalje održavaju prostorni odnosi npr.



Slika 5. Prostorni odnosi se održavaju bez obzira na smanjivanje rezolucije

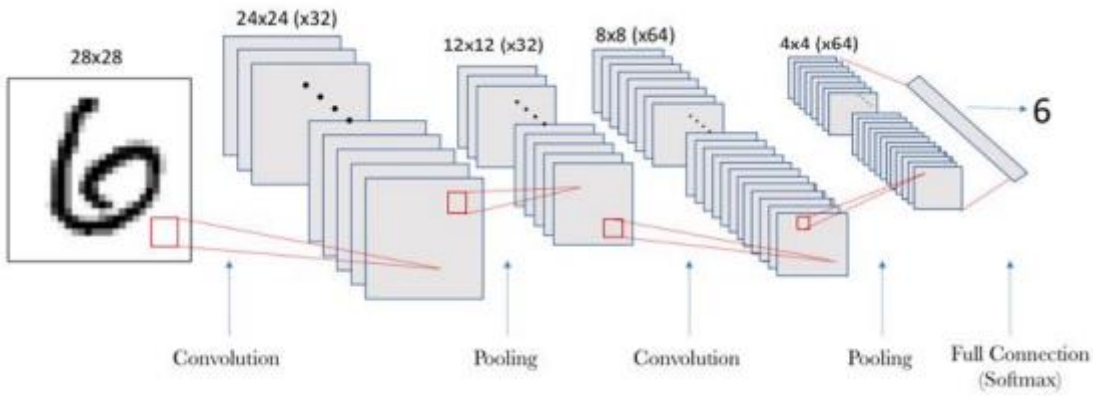
U primjeru prikazanom na Slici 5, za 12 x 12 ulaz i prozor od 2 x 2, na kraju se dobije 6 x 6 izlaz na kojemu se i dalje može razaznati da se radi o broju 7 iako smo time znatno smanjili količinu informacija koju je potrebno obraditi. Kao što je prethodno navedeno, konvolucijski

slojevi imaju više filtera odnosno više podslojeva, te je za svaki taj podsloj individualno potrebno provesti sažimanje.



Slika 6. Sažimanje se provodi za svaki podsloj konvolucijskog sloja individualno

Na kraju konvolucijske neuronske mreže se nalazi potpuno povezani sloj koji predstavlja jedan niz neurona gdje je svaki neuron povezan s svim neuronima prethodnog sloja na sličan način kao što se to radi i kod običnih neuronskih mreža. Nakon takvog sloja se nalazi još jedan sloj koji ima onoliko neurona koliko ima i klasa ulaznih podataka (npr. ako raspoznajemo znamenke onda ima 10 neurona, za znamenke od 0..9) nakon čega se najčešće nekom softmax metodom određuje izlaz iz neuronske mreže.



Slika 7. Kompletna implementacija jedne konvolucijske neuronske mreže

Na slici iznad se može vidjeti tipična konvolucijska neuronska mreža s višestrukim konvolucijskim slojevima i slojevima sažimanja.

PROBLEM RASPOZNAVANJA ZNAMENAKA

Neuronska mreža mora primiti sliku rukom pisane znamenke i prepoznati o kojoj je znamenki riječ. Potrebno je neuronsku mrežu kreirati sa parametrima koji će nam dati najbolje rezultate odnosno koji će osigurati da na glavnoj dijagonali konfuzijske matrice budu vrijednosti što bliže jedinici.

Opis problema

Na ulaz neuronske mreže šalju se skupovi podataka za treniranje i testiranje. Ti se podaci provlače kroz mrežu te određuju vrijednost težine čvorova i filtera. Unaprijed se znaju rezultati koji se moraju dobiti stoga se računa pogreška dobivena iz istih i na osnovu pogreške se opet mijenjaju parametri mreže kako bi se u konačnosti dobili što točniji rezultati. Taj se postupak ponavlja sve do određenog broja iteracija. Sa svakom novom iteracijom neuronska mreža postaje točnija.

Razlog za korištenje konvolucijske neuronske mreže

Standardna neuronska mreža zahtjeva da svaki neuron bude povezan sa svim elementima iz prethodnog sloja stoga ona nije pogodna za analizu slika. Slika 50x50 ima 2500 piksela i rezultirala bi s 2500 težine koju je potrebno podesiti i optimizirati za samo jedan neuron. Konvolucijske mreže zbog svoje arhitekture omogućavaju raspoznavanje uzoraka različitih dimenzija uz puno manje podešavanja i optimiziranja parametara. Svrha CNN-a je da se kroz slojeve neuronske mreže uče raspoznavati inkrementalno kompleksniji uzorci.

PROGRAMSKI KOD

Prvi korak je učitavanje podataka za treniranje i testiranje metodom *load_data()* od MNIST-a. Zatim su podaci mapirani iz intervala 0-255 u interval 0-1. Učitavanje i priprema podataka prikazana je na slici 8.

```
(x_train, y_train), (x_test, y_test1) = mnist.load_data()
x_train = x_train.reshape(60000, 28, 28, 1)
x_train = x_train.astype('float32') / 255
x_test = x_test.reshape(10000, 28, 28, 1)
x_test = x_test.astype('float32') / 255
y_train = utils.to_categorical(y_train, num_classes=10)
y_test = utils.to_categorical(y_test1, num_classes=10)
classes = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Slika 8. Priprema podataka za treniranje konvolucijske mreže

Drugi korak je kreiranje dva linearna niza slojeva odnosno dva Sequential modela. Prvi model sadrži dva konvolucijska sloja, prvi sa 64 filtera, a drugi s 32 filtera. Drugi model također sadrži dva konvolucijska sloja s istim brojem filtera, ali ima i dva sloja za sažimanje. Jedan je između konvolucijskih slojeva, a drugi na kraju.

```
#PRVI
def first_CNN(activation_f='relu', kernel_size=(3, 3)):
    model = models.Sequential()

    model.add(layers.Conv2D(64, kernel_size, activation=activation_f, input_shape=(28, 28, 1),

    model.add(layers.Conv2D(32, kernel_size, activation=activation_f, strides=(1, 1)))

    model.add(layers.Flatten())
    model.add(layers.Dense(10, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

    return model

#DRUGI
def second_CNN(activation_f='relu', kernel_size=(3, 3)):
    model = models.Sequential()

    model.add(layers.Conv2D(64, kernel_size, activation=activation_f, input_shape=(28, 28, 1),
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    model.add(layers.Conv2D(32, kernel_size, activation=activation_f, strides=(1, 1)))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    model.add(layers.Flatten())
    model.add(layers.Dense(10, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

    return model
```

Slika 9. Izrada CNN modela

Kreirana je funkcija *ConfusionMatrix()* koja kreira i sprema konfuzijsku matricu. Kao što je već spomenuto cilj je da rezultati na glavnoj dijagonali konfuzijske matrice budu što bliži jedinici.

```
def ConfusionMatrix(confusionMatrixdf, path):
    plt.figure()
    sns.heatmap(confusionMatrixdf, annot=True, cmap=plt.cm.Blues)
    plt.tight_layout()
    plt.ylabel("True label")
    plt.xlabel("Predicted label")
    title_temp = path.split('/')[-1].split('_')
    title = "Activation fnc: " + title_temp[0] + " -- Kernel size: " + title_temp[1]
    plt.title(title)
    plt.tight_layout()
    plt.savefig(path + ".png")
    plt.close()
```

Slika 10. Funkcija za spremanje konfuzijske matrice

Preko for petlje je ostvareno to da se prolazi kroz različite vrijednosti za veličinu jezgre i aktivacijske funkcije te za sve kombinacije kreira i uči mrežu kako bi se utjecaji parametara mogli usporediti. Isto je napravljeno za oba CNN modela.

```
model1HeatmapData = np.zeros((len(kernel_sizes), len(activation_functions)))
i = 0
for act_f in activation_functions:
    j = 0
    for kernel in kernel_sizes:
        first_model = first_CNN(activation_f=act_f, kernel_size=kernel)
        first_model.summary()
        first_model.summary(print_fn=lambda x: logger1.write(x + '\n'))
        history = first_model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5)
        logger1.write('\n'.join('{}: {}'.format(x[0], x[1]) for x in history.history))
        logger1.write('\n\n\n')
        y_pred = first_model.predict_classes(x_test)
        cm = confusion_matrix(y_test1, y_pred)
        cm_norm = np.around(cm.astype('float') / cm.sum(axis=1)[:, np.newaxis], decimals=2)
        cm_norm_df = pd.DataFrame(cm_norm, index=classes, columns=classes)
        path_dir, path_file = createModelPathAndName(model=1, activ_f=act_f, kernel_size=kernel)
        mkdir_p(path_dir)
        ConfusionMatrix(confusionMatrixdf=cm_norm_df, path=path_file)

        model1HeatmapData[i][j] = accuracy_score(y_test1, y_pred)
        j += 1
    i += 1
```

Slika 11. Učenje prvog modela

Standardna neuronska mreža je preuzeta s prethodne laboratorijske vježbe s izmjenom u podacima za treniranje i učenje. Ovaj puta je potrebno koristiti MNIST bazu podataka. Klasična neuronska mreža je kreirana preko *MLPClassifier* klase sa određenim parametrima.

Učenje klasične neuronske mreže napravljeno je *fit()* funkcijom kojoj su predani skupovi podataka za učenje. Testiranje je napravljeno *predict()* funkcijom. Konfuzijskom matricom prikazana je točnost mreže.

```
X,Y = fetch_openml('mnist_784',version=1,return_X_y=True)
X=X/255

X_train, X_test=X[:60000],X[60000:]
Y_train, Y_test=Y[:60000],Y[60000:]

mlp = MLPClassifier(hidden_layer_sizes=(100),activation='logistic',solver='adam', alpha=0.0001,max_iter = 100)

with warnings.catch_warnings():
    warnings.filterwarnings("ignore",category=ConvergenceWarning,module='sklearn')
    mlp.fit(X_train, Y_train)

print("Training set score: %f" % mlp.score(X_train,Y_train))
print("Test set score: %f" % mlp.score(X_test,Y_test))
score = mlp.score(X_test,Y_test)

y_pred = mlp.predict(X_test)
cm = metrics.confusion_matrix(Y_test,y_pred)
cm_norm=np.around(cm.astype('float')/cm.sum(axis=1)[:,np.newaxis],decimals=2)
sns.heatmap(cm_norm, annot=True)
plt.show()
```

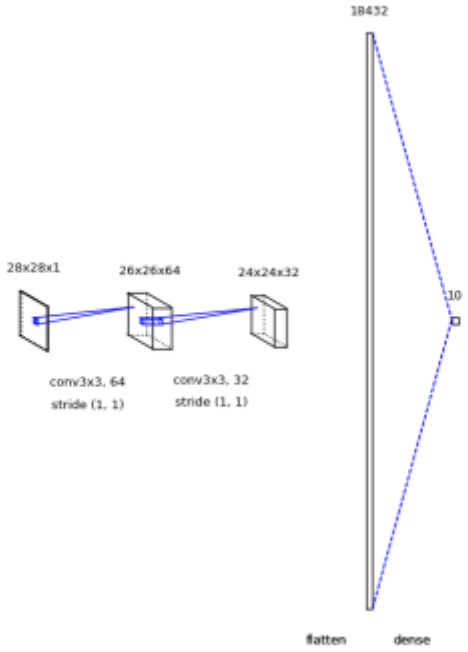
Slika 12. Standardna neuronska mreža

RJEŠENJE

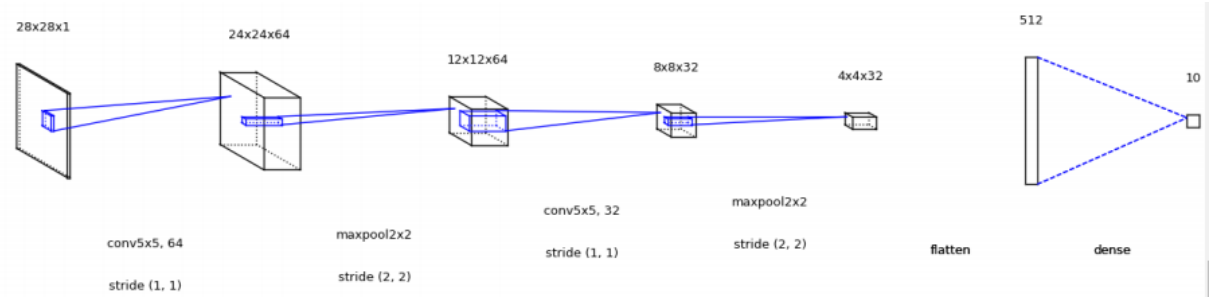
Zadatak

Potrebno je projektirati i ispitati konvolucijsku neuronsku mrežu koja će naučiti raspoznavati znamenke. Za treniranje i testiranje koristiti MNIST bazu, a za implementaciju koristiti Python programski jezik i Keras biblioteku s pripadajućim Tensorflow backend-om. MNIST baza sadrži 70000 slika znamenaka od kojih se 60000 koristi za treniranje, a 10000 za testiranje

Prvi model treba implementirati da bude kao na slici 13, a drugi kao na slici 14.



Slika 13. Prvi model konvolucijske neuronske mreže



Slika 14. Drugi model konvolucijske neuronske mreže

Kao što se može vidjeti, prvi model sadrži samo dva konvolucijska sloja, prvi sa 64 filtera, a drugi s 32 filtera. Drugi model isto ima dva konvolucijska sloja s istim broj filtera kao i prethodni model, ali za razliku od prethodnog modela ima i dva sloja za sažimanje, jedan između konvolucijskih slojeva i jedan na kraju, prije potpuno povezanog modela. Slojevi za sažimanje koriste max funkciju i veličinu prozora 2 x 2.

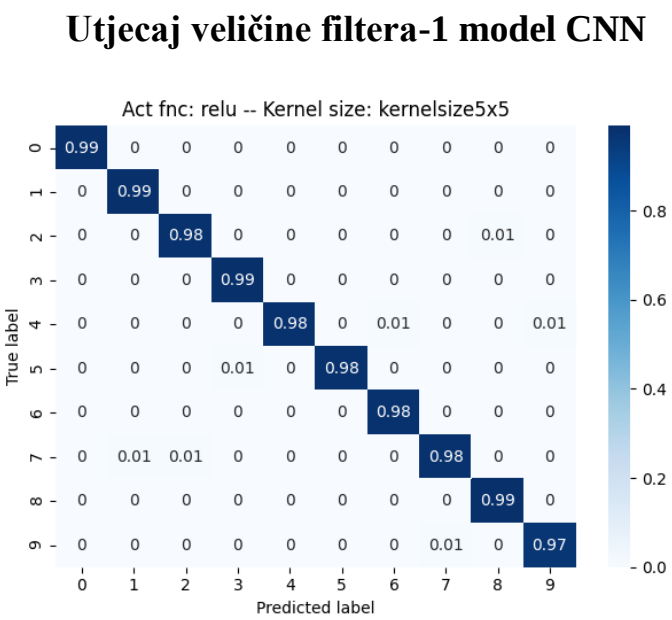
Implementirati dva navedena modela i mijenjati im parametre na slijedeći način:

- Aktivacijska funkcija: 'relu', 'tanh', 'sigmoid'
- Veličina filtera (kernel-a) konvolucijskih slojeva: 3 x 3, 5 x 5, 7 x 7

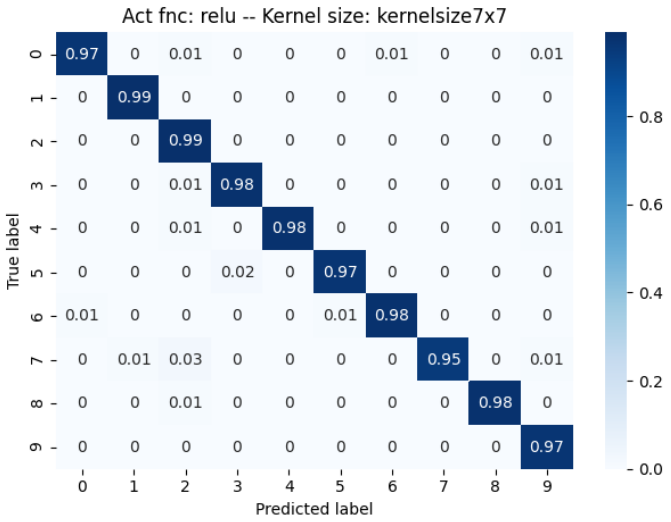
Učiti stvorene modele kroz najviše 5 epoha.

Usporediti performanse sa standardnom neuronskom mrežom za isti problem, raspoznavanje znamenaka u MNIST bazi, korištenjem scikit-learn biblioteke. Implementacija standardne neuronske mreže treba imati 100 neurona, koristiti 'logistic' aktivacijsku funkciju, 'adam' optimizacijski postupak i prolaziti kroz najviše 100 epoha za učenja. Za ovaj dio vježbe koristiti skriptu NN_classification_s.py koja je dostupna za LV5 (Klasificiranje vrste cvijeta Irisa).

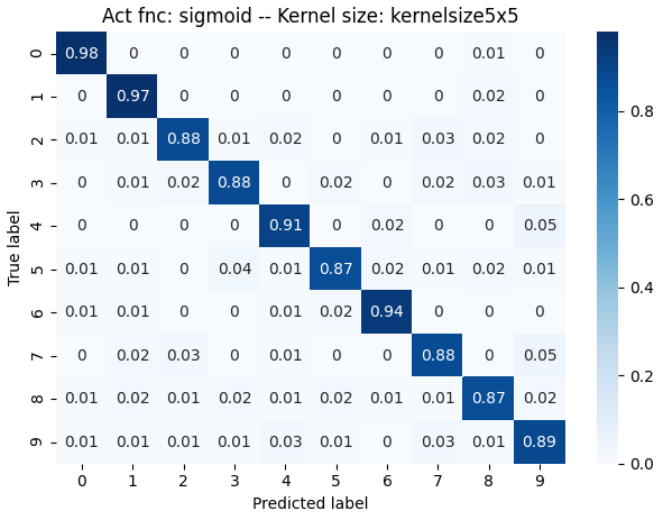
Rezultati



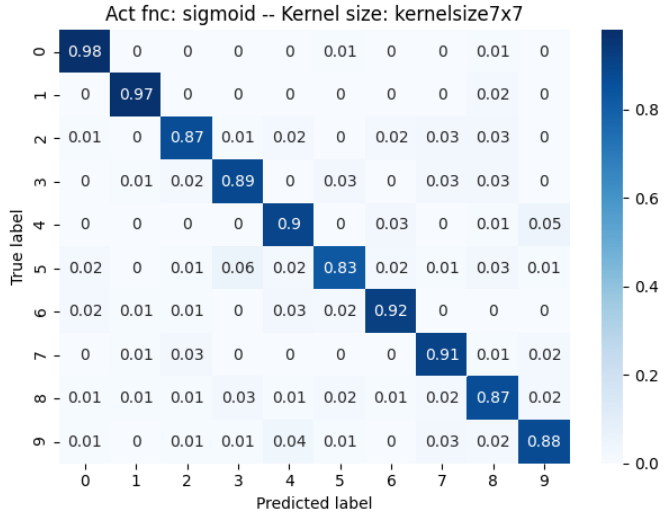
Slika 15. Aktivacijska funkcija 'relu', veličina filtera 5x5



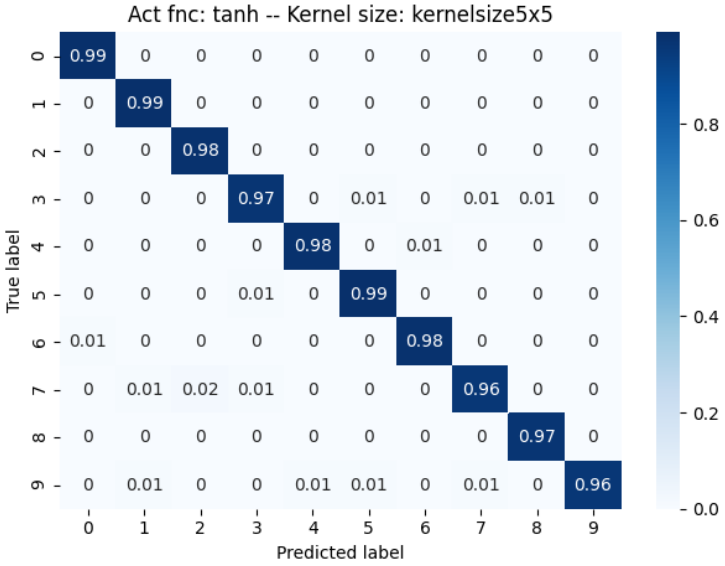
Slika 16. Aktivacijska funkcija 'relu', veličina filtera 7x7



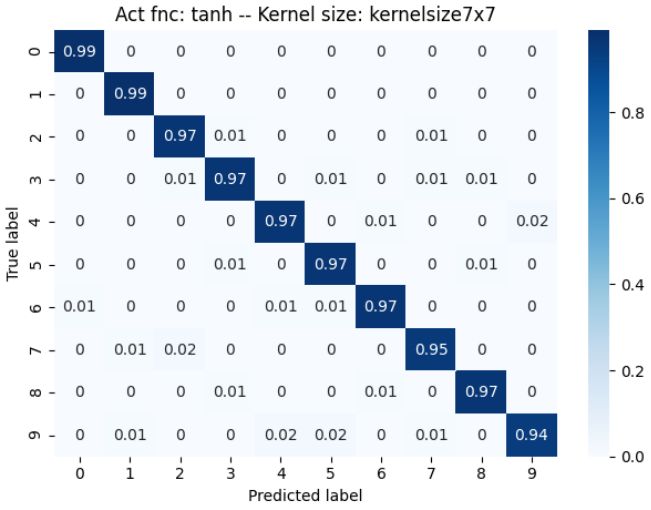
Slika 17. Aktivacijska funkcija 'sigmoid', veličina filtera 5x5



Slika 18. Aktivacijska funkcija 'sigmoid', veličina filtera 7x7



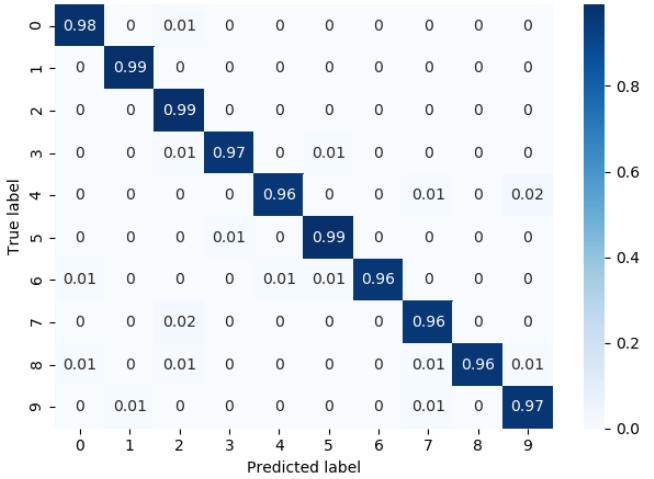
Slika 19. Aktivacijska funkcija 'tanh', veličina filtera 5x5



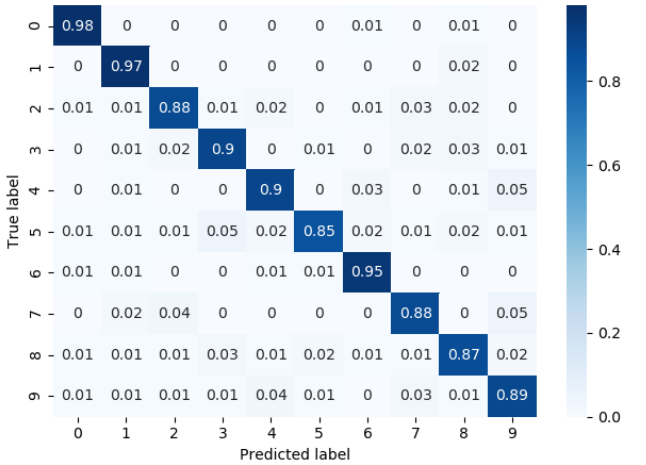
Slika 20. Aktivacijska funkcija 'tanh', veličina filtera 7x7

Utjecaj veličine filtera-2 model CNN

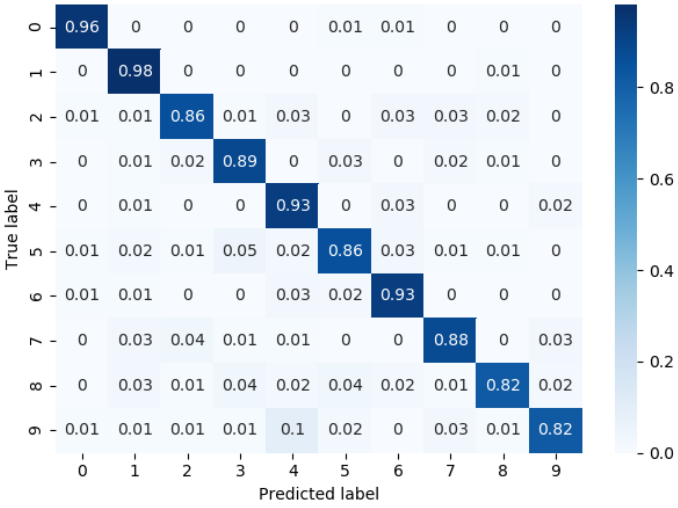
Slika 21. Aktivacijska funkcija 'relu', veličina filtera 5x5



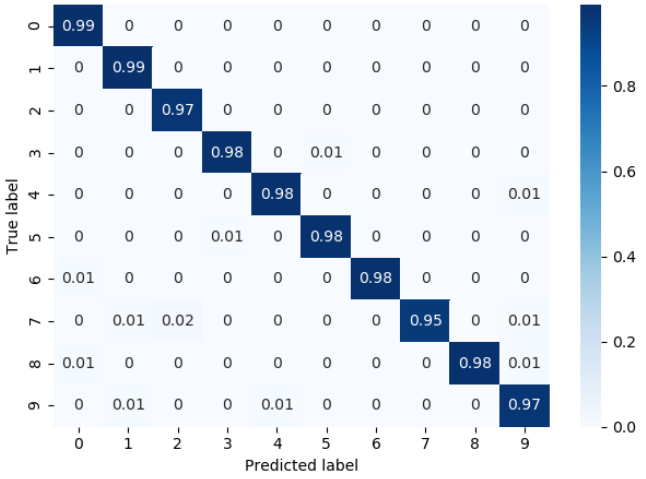
Slika 22. Aktivacijska funkcija 'relu', veličina filtera 7x7



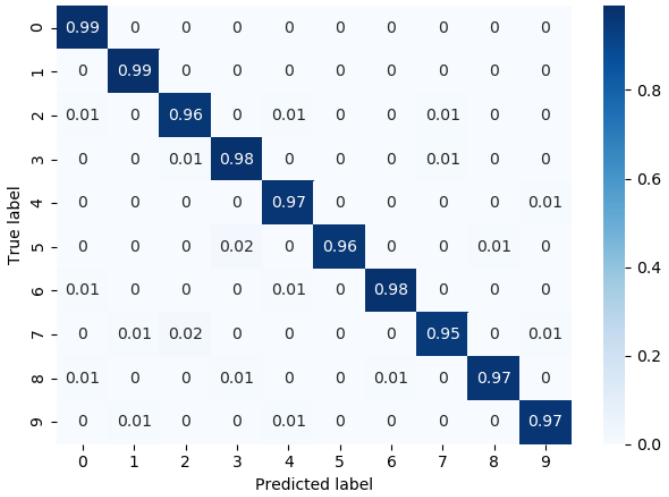
Slika 23. Aktivacijska funkcija 'sigmoid', veličina filtera 5x5



Slika 24. Aktivacijska funkcija 'sigmoid', veličina filtera 7x7

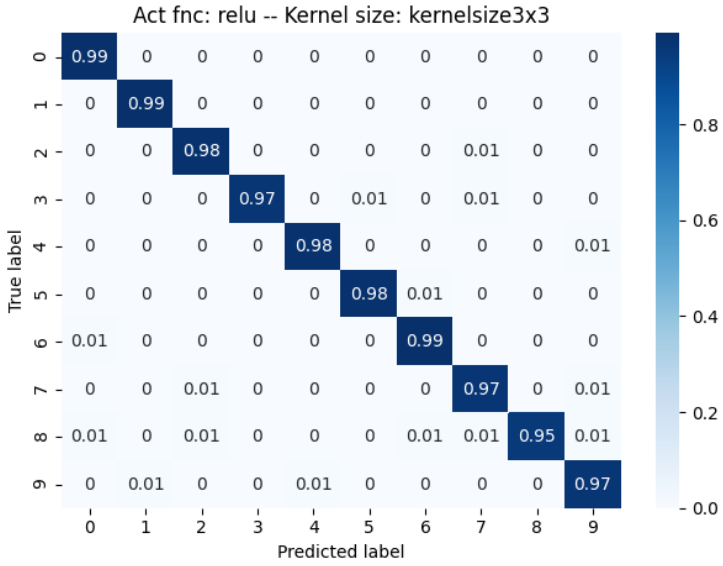


Slika 25. Aktivacijska funkcija 'tanh', veličina filtera 5x5

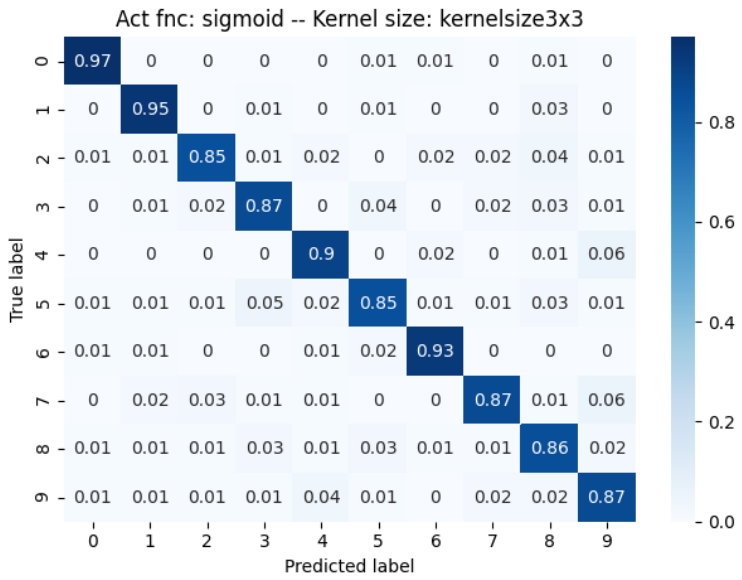


Slika 26. Aktivacijska funkcija 'tanh', veličina filtera 7x7

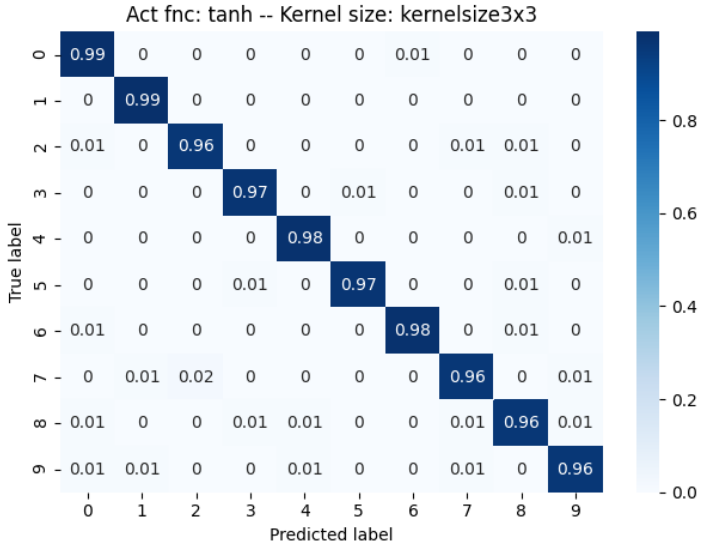
Utjecaj aktivacijske funkcije skrivenog sloja-1 model CNN



Slika 27. Aktivacijska funkcija 'relu', veličina filtera 3x3

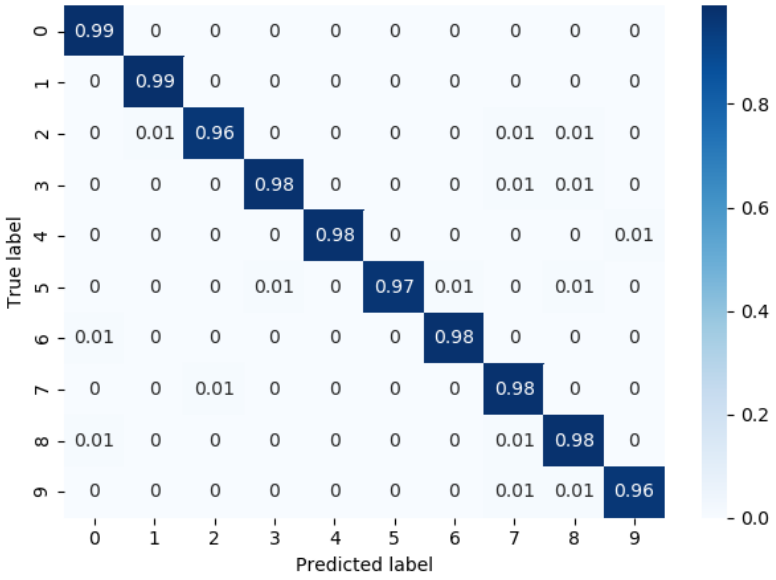


Slika 28. Aktivacijska funkcija 'sigmoid', veličina filtera 3x3

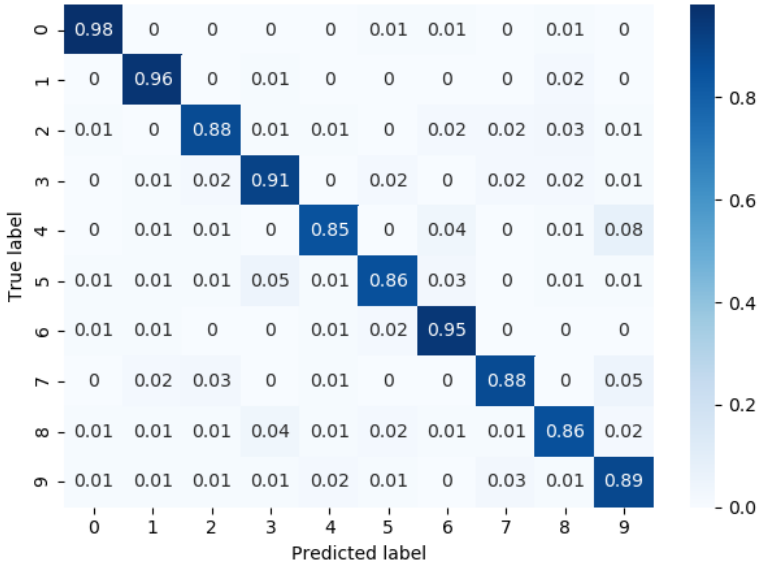


Slika 29. Aktivacijska funkcija 'tanh', veličina filtera 3x3

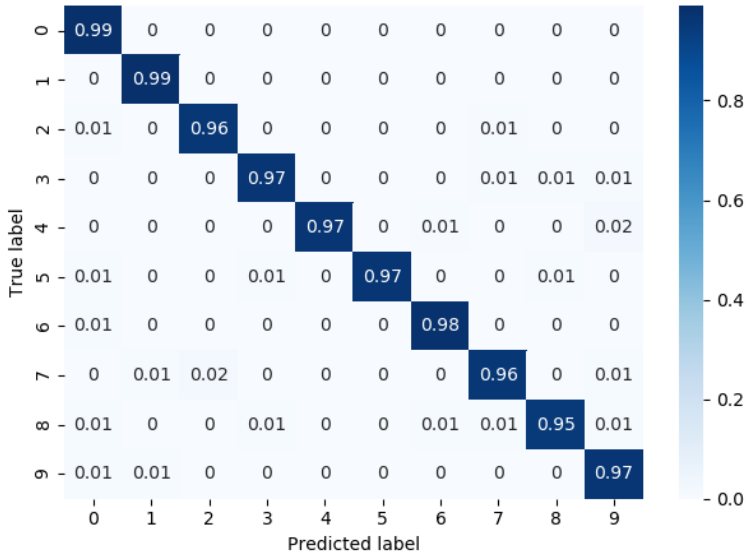
Utjecaj aktivacijske funkcije skrivenog sloja-2 model CNN



Slika 30. Aktivacijska funkcija 'relu', veličina filtera 3x3

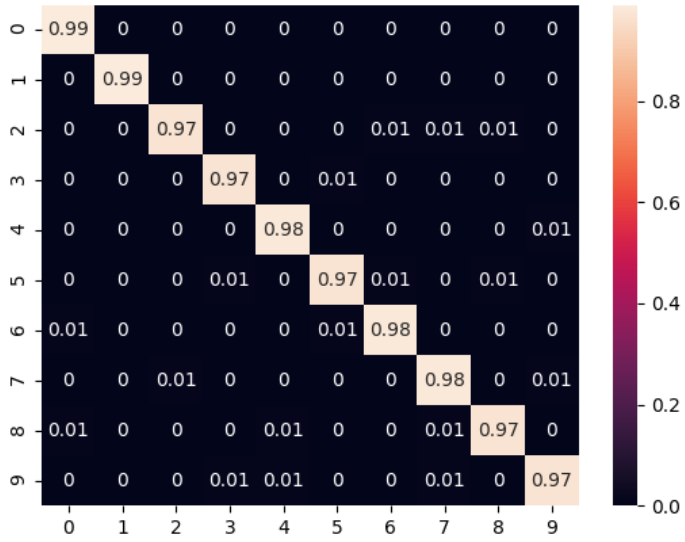


Slika 31. Aktivacijska funkcija 'sigmoid', veličina filtera 3x3



Slika 32. Aktivacijska funkcija 'tanh', veličina filtera 3x3

Standardna neuronska mreža



Slika 33. Standardna neuronska mreža

ZAKLJUČAK

U šestoj laboratorijskoj vježbi kreirane su dva modela konvolucijskih neuronskih mreža i jedna standardna neuronska mreža. Sve neuronske mreže su imale isti zadatak to jest morale su raspoznati o kojem je broju riječ na slici na kojoj se nalazio rukom pisan broj. Kao baza podataka za treniranje i ispitivanje korištena je MNIST baza podataka. Prilikom uspoređivanja mijenjani su parametri kako bi se utvrdio utjecaj istih na mrežu. Za standardne neuronske mreže se u startu zna da nisu pogodne u svrhu analize slika zbog podešavanja težine dok konvolucijske neuronske mreže zbog arhitekture omogućavaju raspoznavanje uzoraka različitih dimenzija uz puno manje podešavanja parametara i optimiziranja.

Iz rezultata je vidljivo da aktivacijska funkcija 'relu' daje najtočnije vrijednosti dok su najlošije sa aktivacijskom funkcijom 'sigmoid'.

Smanjenjem veličine filtera dobivamo lošiji rezultat.

Rezultati standardne neuronske mreže su dobri, no vrijeme učenja je znatno veće nego kod konvolucijskih mreža. Razlog tome je što njihovo vrijeme izvođenja ovisi o dimenziji fotografije.