Neuronske mreže:
Prepoznavanje novčanica

Dimnjaković Iva
Gorup Lovro
Lovrić Filipa

Sveučilište u Zagrebu
Prirodoslovno – matematički fakultet
Matematički odsjek

2022./2023.

Sadržaj

Sadržaj 1
1. Uvod 2
2. Prednosti neuronskih mreža u prepoznavanju novčanica 3
3. Općenito o neuronskim mrežama 4
4. Konvolucijske neuronske mreže 7
4.1. Što je konvolucijska neuronska mreža 7
4.2. Primjene konvolucijskih neuronskih mreža 8
4.3. Alati za programiranje konvolucijskih mreža 8
4.4. Nedostaci konvolucijskih mreža 9
4.5. Budućnost korištenja konvolucijskih mreža 9
4.6. Učitavanje podataka 10
5. Priprema skupa podataka 11
6. Pretprocesiranje podataka 13
7. Računanje točnosti 13
8. Implementacija neuronske mreže 14
8.1. Treniranje neuronske mreže 16
8.2. Testiranje neuronske mreže 17
9. KNN klasifikator 18
10. SVC klasifikator 20
11. Greške u konfuzijskim matricama 22
12. Zaključak 23
13. Literatura 24

1. Uvod

Prepoznavanje novčanica važan je zadatak u bankarskom sektoru, trgovačkim automatima, bankomatima, a također i u različitim drugim aplikacijama. Korištenjem neuronskih mreža moguće je automatizirati ovaj postupak prepoznavanja novčanica što daje značajnu uštedu vremena i novca te se smanjuje potencijalna greška koju bi mogli napraviti ljudi.

Neuronske mreže su računalni modeli koji se temelje na biološkim neuronima i sinapsama. One se sastoje od slojeva neurona koji obrađuju ulaze i donose izlaze. Ti modeli su sposobni naučiti prepoznavati uzorke i klasificirati ih u kategorije.

U ovom radu će se najprije objasniti općeniti rad neuronske mreže (s naglaskom na konvolucijski tip neuronskih mreža) i njezine prednosti u odnosu na ostale načine klasifikacije. Zatim će se implementirati KNN i SVC klasifikatori za primjer novčanica te će se konstruirati konvolucijska neuronska mreža za isti primjer. Na kraju će se rezultati klasifikacije međusobno usporediti.

2. Prednosti neuronskih mreža u prepoznavanju novčanica

Brojne su prednosti korištenja neuronskih mreža u prepoznavanju novčanica:

1. Brzina i točnost - neuronske mreže su sposobne brzo obraditi velike količine podataka u kratkom vremenskom razdoblju, što ih čini vrlo učinkovitim za prepoznavanje novčanica. Također, kada se pravilno treniraju, neuronske mreže mogu postići visoku točnost prepoznavanja novčanica.

2. Automatizacija - upotreba neuronskih mreža omogućuje potpunu automatizaciju postupka prepoznavanja novčanica, što značajno smanjuje potencijalne pogreške koje bi mogli napraviti ljudi.

3. Fleksibilnost - neuronske mreže mogu se prilagoditi različitim vrstama novčanica i različitim valutama. To ih čini vrlo fleksibilnim i korisnim za širok raspon aplikacija.

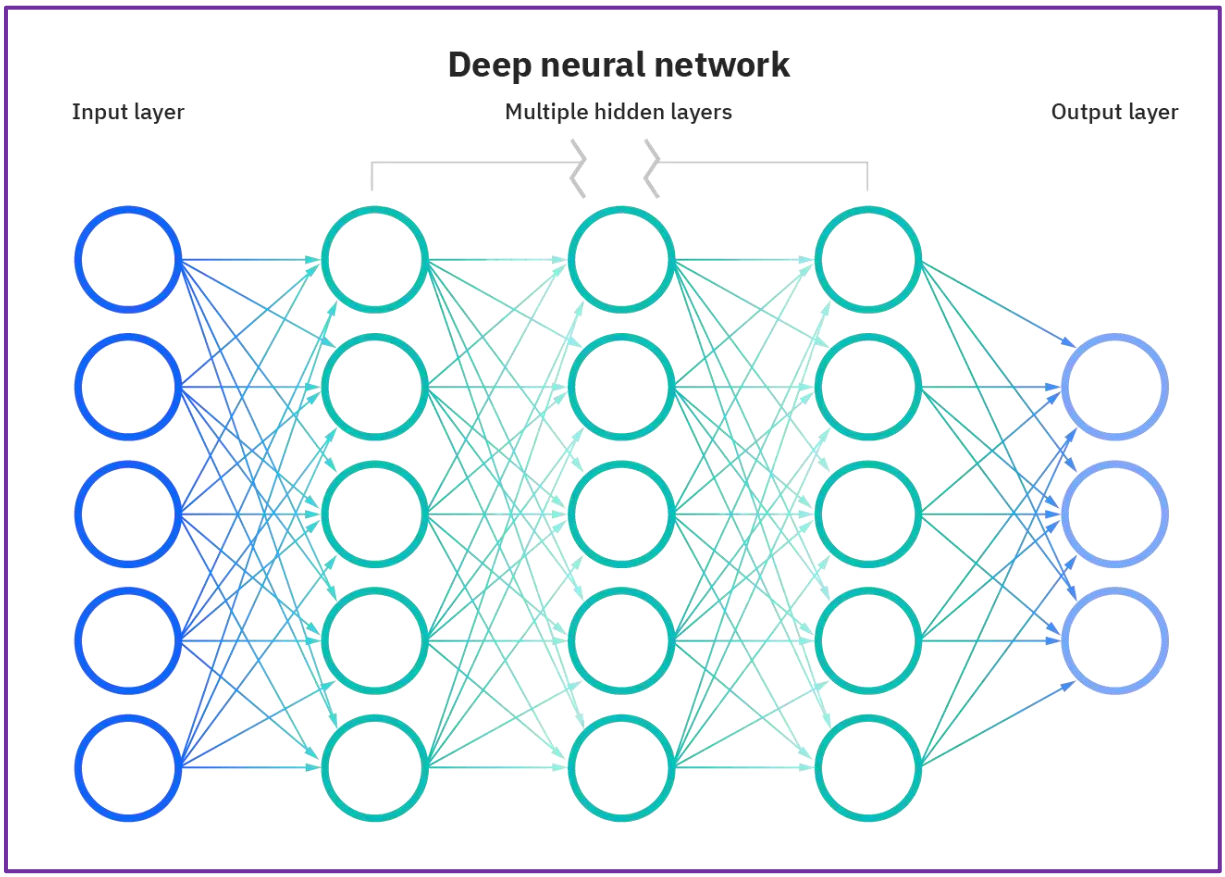
4. Skalabilnost - neuronske mreže mogu se skalirati kako bi se mogli obrađivati veliki brojevi novčanica.

5. Učenje na temelju iskustva - neuronske mreže mogu naučiti prepoznavati nove novčanice koje nisu bile prisutne u skupu podataka za treniranje. To se postiže kontinuiranim učenjem na temelju iskustva.

3. Općenito o neuronskim mrežama

Osnovni objekt neuronske mreže je neuron (ili čvor), po čemu je i mreža dobila ime. Skup neurona, od kojih svaki ima pridijeljenu vrijednost iz skupa [0, 1], međusobno je povezan bridovima koji imaju određenu težinu. Pri tome, apsolutna vrijednost između dva neurona označava snagu povezanosti (što je apsolutna vrijednost veća, to je veza snažnija), dok predznak označava kako neuroni djeluju („pobuđuju“ li se ili obratno).

Nadalje, mreža neurona podijeljena je u nekoliko slojeva. Prvi i zadnji sloj nazivaju se ulazni i izlazni sloj te postoje još proizvoljno slojeva između koji se skupno nazivaju skriveni sloj.



Slika Shematski prikaz neuronske mreže

Na gornjoj slici neuroni su prikazani krugovima, a njihove veze usmjerenim linijama te jedan vertikalni niz neurona čini jedan sloj.

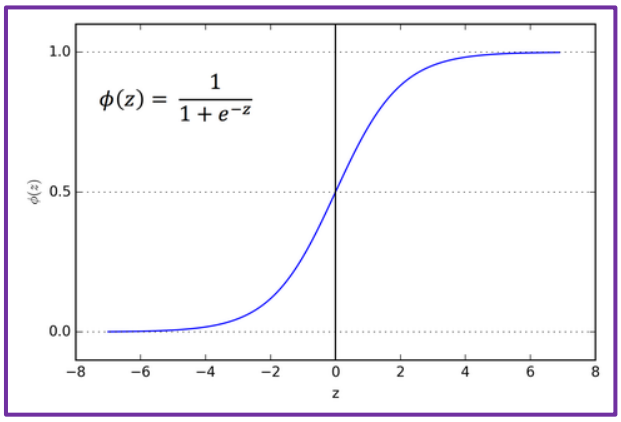
Opišimo rad neuronske mreže. Na početku su zadane samo vrijednosti čvorova ulaznoga sloja. Najprije je potrebno pridijeliti inicijalne vrijednosti ostalim slojevima.

Označimo s a_0, … , a_n vrijednosti čvorova prvog sloja, b_0, … , b_m (zasad još nedefinirane) vrijednosti čvorova drugog sloja te s $w_{0,0}$, … , $w_{n,m}$ vrijednosti težina između dva neurona iz prvog i drugog sloja.

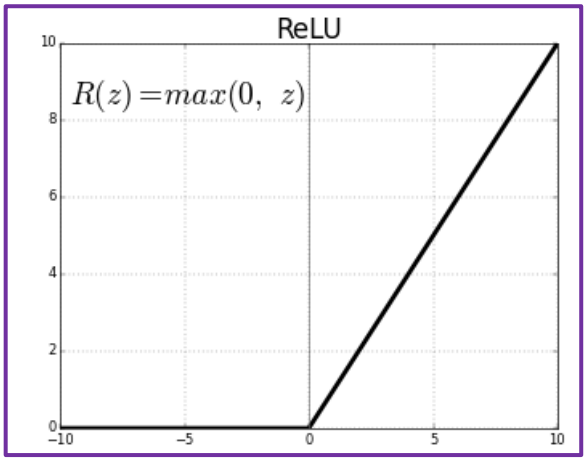
Tada za čvor b_0 računamo sljedeću linearnu kombinaciju:

$$a_0w_{0,0} + \ldots + a_nw_{n,0} + b,$$

gdje je b tzv. *bias*, koji služi za pomicanje vrijednosti čvora u lijevo ili desno. Time dobivamo neku realnu vrijednost, no potrebno je transformirati tu vrijednost da bude u skupu [0, 1]. To se dobiva primjenom određene funkcije koje se nazivaju aktivacijske funkcije, npr. funkcije *sigmoid* ili *ReLu* funkcije.



Slika Sigmoid funkcija



Slika ReLu funkcija

Analogno postupamo za sve čvorove drugog sloja te analogno za svaki sloj do predzadnjeg sloja čime smo pridijelili svakom čvoru, osim u zadnjem sloju, vrijednost. Za zadnji sloj ne primjenjujemo aktivacijske funkcije jer je potrebno da izlazne vrijednosti budu probabilističke, tj. da suma svih vrijednosti čvorova izlaznoga sloja bude 1. To se može postići npr. primjenom funkcija *softmax* ili *logsoftmax*.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Slika Softmax funkcija

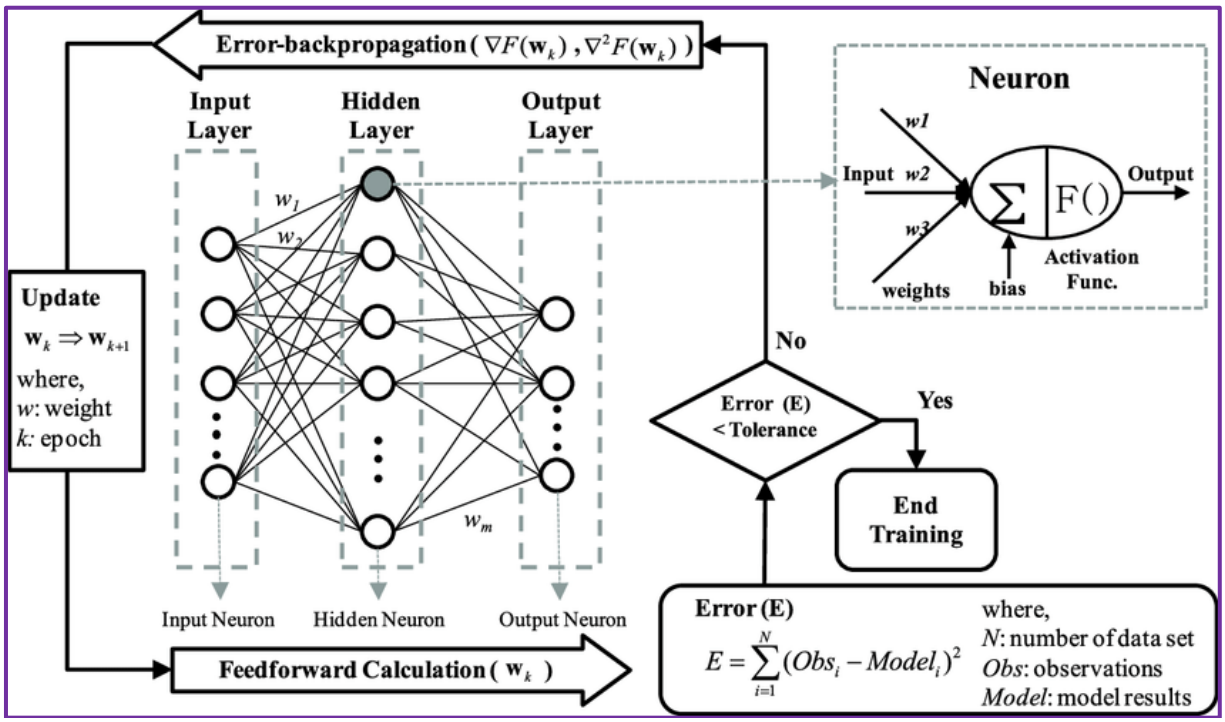
$$\mathrm{LogSoftmax}(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right)$$

Slika Logsoftmax funkcija

Nakon toga su u izlaznom sloju spremljene vrijednosti čvorova koji sadrže vjerojatnosti svakog rezultata, ali općenito nakon prvog popunjavanja nemaju nikakve veze s njim. Stoga, potrebno je provesti postupak koji se naziva treniranje neuronske mreže.

Treniranje radi na skupu podataka za koji se zna koji su očekivani rezultati. Za svaki podatak iz skupa se provede gornji postupak i dobiveni rezultat se usporedi s očekivanim. To omogućuje ispravljanje neuronske mreže, točnije mijenjaju se vrijednosti težina bridova između neurona na način da se smanji greška između rezultata u danom primjeru. Postizanje toga se svodi na minimiziranje funkcije greške (koja ovisi o parametrima težina), a znamo iz linearne algebre da se za to koristi gradijent.

Gore opisani postupak se naziva *backpropagation*, odnosno propagiranje unatrag. Time se dobiva neuronska mreža koja u određenoj točnosti, u ovisnosti o broju primjeraka korištenih za treniranje i prirode samog problema, klasificira nove, testne primjerke, u odgovarajuće klase.



Slika 6 Shematski prikaz propagiranja unatrag

4. Konvolucijske neuronske mreže

4.1. Što je konvolucijska neuronska mreža

Konvolucijska neuronska mreža (CNN) je vrsta neuronske mreže koja se najčešće koristi za obradu slika, prepoznavanje uzoraka i druge vrste analize podataka. Ova tehnologija postala je sve popularnija u posljednjih nekoliko godina zbog svoje sposobnosti da prepozna uzorke koji su nevidljivi ljudskom oku i da identificira objekte i druge karakteristike na temelju tih uzoraka.

Konvolucijske neuronske mreže su nadogradnja tradicionalnih neuronskih mreža koje se koriste za prepoznavanje uzoraka u slikama. Za razliku od tradicionalnih neuronskih mreža koje se temelje na potpuno povezanim slojevima, konvolucijske neuronske mreže se temelje na konvolucijskim slojevima.

Konvolucijski slojevi su slojevi koji primjenjuju konvoluciju na ulazne podatke. Konvolucija je matematička operacija koja omogućuje da se jedan skup podataka transformira u drugi skup podataka. U slučaju konvolucijskih neuronskih mreža, ulazni podaci su slike, a konvolucija se primjenjuje kako bi se stvorile značajke koje se koriste za prepoznavanje objekata na tim slikama.

Konvolucijske neuronske mreže koriste višeslojne neuronske mreže koje se sastoje od slojeva konvolucije, nakon kojih slijedi sloj aktivacije i sloj grupiranja.

U sloju konvolucije, mreža koristi matricu koja se naziva „filter“ ili „jezgra“. Filter se kreće preko slike korak po korak i izračunava se dotproizvod piksela slike i filtera. Rezultat ovog dotproizvoda predstavlja aktivaciju pojedinih neurona u sljedećem sloju. Sloj konvolucije obično se sastoji od više filtera, što omogućuje da se detektiraju različiti uzorci i značajke na slici.

Sloj aktivacije obično slijedi sloj konvolucije. Ovaj sloj primjenjuje aktivacijsku funkciju na izlaz iz sloja konvolucije. Aktivacijska funkcija služi za dodavanje nelinearnosti u mrežu, što omogućuje da mreža nauči složenije značajke i uzorke.

Sloj grupiranja se koristi za smanjivanje dimenzija izlaza iz sloja aktivacije. Ovaj sloj koristi matricu za grupiranje piksela aktivacije, što omogućuje da se smanji

količina podataka koju mreža obrađuje u sljedećem sloju. Sloj grupiranja se obično koristi nakon sloja aktivacije kako bi se dodatno poboljšala sposobnost mreže da izdvoji ključne značajke iz slike.

4.2. Primjene konvolucijskih neuronskih mreža

Konvolucijske neuronske mreže su pronašle primjenu u mnogim područjima, uključujući prepoznavanje lica, prepoznavanje emocija, detekciju objekata u stvarnom vremenu, prepoznavanje govora i rukom pisanih slova i brojeva. Sve više tvrtki i istraživačkih institucija koristi konvolucijske neuronske mreže za poboljšanje svojih proizvoda i usluga.

Jedna od najpopularnijih primjena konvolucijskih neuronskih mreža je klasifikacija slika.

Kod klasifikacije slika, mreža se trenira na skupu slika koje su povezane s određenim klasama. Skup slika se naziva "skup za učenje" (engl. training set) i koristi se za treniranje mreže. Nakon što je mreža trenirana, može se koristiti za klasifikaciju novih slika - mreža analizira sliku i određuje kojoj klasi slika pripada.

4.3. Alati za programiranje konvolucijskih mreža

Uz sve veću primjenu konvolucijskih neuronskih mreža u različitim područjima, postoji i sve više dostupnih alata za programiranje konvolucijskih neuronskih mreža - TensorFlow, PyTorch, Keras i Caffe. Ovi alati omogućuju programerima da brzo i jednostavno izgrade i treniraju svoje mreže. Najpopularniji alat za programiranje konvolucijskih neuronskih mreža je TensorFlow - otvoreni izvorni kod koji je razvila tvrtka Google. TensorFlow je razvijen kako bi bio lako korišten i fleksibilan alat za programiranje neuronskih mreža, uključujući konvolucijske neuronske mreže.

Korištenje konvolucijskih neuronskih mreža može biti izazovno zbog velikog broja parametara koji se koriste za optimizaciju. Međutim, s pravilnim programiranjem i korištenjem dostupnih alata, konvolucijske neuronske mreže mogu biti vrlo učinkovit način za obradu slika i prepoznavanje uzoraka.

4.4. Nedostaci konvolucijskih mreža

Uz sve prednosti, konvolucijske neuronske mreže imaju i neke nedostatke. Jedan od njih je njihova velika složenost. Velik broj parametara i slojeva mreže zahtijeva veliku količinu resursa, kao što su procesorsko vrijeme i memorija. Stoga, implementacija i treniranje konvolucijskih neuronskih mreža može biti prilično skupo.

Drugi nedostatak konvolucijskih neuronskih mreža je njihova osjetljivost na veličinu ulaznih podataka. Kako su mreže izgrađene za specifične dimenzije ulaznih podataka, promjena veličine ulaza može dovesti do pogrešaka u izlazu mreže.

Također, konvolucijske neuronske mreže nisu uvijek transparentne, što znači da je teško razumjeti kako se donose odluke i koji su dijelovi ulaznih podataka najvažniji za donošenje odluka. Ovaj nedostatak može biti kritičan u nekim primjenama, kao što su medicina i pravosuđe, gdje je potrebno pružiti jasno obrazloženje odluka koje donosi algoritam.

Unatoč navedenim nedostacima, konvolucijske neuronske mreže su nevjerojatno moćan alat za analizu podataka. Oni su pomogli da se postignu značajni napretci u području računalnog vida, prepoznavanja uzoraka, obrade slike i drugih područja strojnog učenja. Konvolucijske neuronske mreže se i dalje razvijaju i poboljšavaju, čime se otvaraju nova područja primjene i stvaraju se nove mogućnosti za automatizaciju i optimizaciju poslovnih procesa.

4.5. Budućnost korištenja konvolucijskih mreža

Konvolucijske neuronske mreže su u posljednjih nekoliko godina postale vrlo popularne u području umjetne inteligencije. Mnogi ljudi vjeruju da će konvolucijske neuronske mreže biti ključne za razvoj autonomnih vozila, robotskih sustava i drugih vrsta umjetne inteligencije koji zahtijevaju analizu složenih podataka, uključujući slike i zvuk.

Zahvaljujući njihovoj sposobnosti da automatski nauče značajke iz složenih podataka, konvolucijske neuronske mreže otvaraju nove mogućnosti u različitim industrijskim sektorima i poslovnim područjima. U budućnosti se

očekuje da će konvolucijske neuronske mreže postati još složenije i naprednije, što će otvoriti nova područja primjene i poboljšati performanse postojećih sustava.

4.6. Učitavanje podataka

Podatke smo učitali kroz generatore kako bi zauzeli manje memorije jer neuronske mreže same po sebi zauzimaju puno memorije. Kod KNN i SVC klasifikatora su svi podaci istovremeno učitani.

5. Priprema skupa podataka

Podaci koji se koriste u implementaciji neuronske mreže su fotografije Bangladeških novčanica vrijednosti 1, 2, 5, 10, 20, 50, 100, 500 i 1000 preuzete s Kaggle-a (link: https://www.kaggle.com/datasets/nsojib/bangla-money) .

Vrijednost: 2

Vrijednost: 20

Vrijednost: 50

Vrijednost: 1000

Slika Primjeri slika iz skupa podataka

U sljedećoj je tablici prikazano koliko je slika novčanica korišteno za trening i za testiranje po klasama (s tim da valja naglasiti da je 20% podataka za trening izdvojeno za validaciju):

Klasa:	1	2	5	10	20	50	100	500	1000
Trening:	101	213	213	213	173	213	208	136	167
Test:	20	37	37	38	37	37	37	37	53

Sve su fotografije dimenzija 120 x 250 te je stoga *preprocessing* obavljen u dva koraka – prvo su se sve fotografije preoblikovale tako da imaju jednu dimenziju (reshape), a zatim im je pomoću funkcije preprocessing.scale oduzeta aritmetička sredina te je dobiveni rezultat podijeljen sa standardnom

devijacijom tako da su u konačnici dobivene vrijednosti slike sa aritmetičkom sredinom 0 i standardnom devijacijom 1.

6. Pretprocesiranje podataka

Kako bi se procijenili parametri modela neuronske mreže potrebno je podijeliti uzorak na dio za treniranje i validaciju u mreži (engl. in-the-sample) te na dio koji je u trenutku procjene parametara ostavljen sa strane, tj. uzorak za testiranje (engl. out-of sample) s ciljem procjene točnosti predviđanja modela neuronske mreže na novim podacima.

Podatke smo učitali pozivom funkcije *load_data* koja čita svaku slikovnu datoteku iz svake mape navedene u *train_folder_paths*, pretvara slikovne podatke u NumPy niz pomoću metode *np.array* iz NumPy biblioteke i dodaje rezultirajući niz u listu *x*. Funkcija također izvlači oznaku klase iz putanje mape pomoću metode dijeljenja i dodaje je na kraj liste *y*. Na kraju funkcija vraća obje liste *x* i *y*.

Nakon učitavanja podataka, koristili smo funkciju *train_test_split* iz *scikit_learn* kako bi na slučajan način podijelili liste *x* i *y* u skupove za treniranje i validaciju, na način da je 80% podataka pridijeljeno skupu za treniranje i 20% podataka pridjeljeno skupu za validaciju.

Zatim smo preoblikovali slikovne podatke u *x*, *x_train*, *x_val* i *x_test* u jednodimenzionalno polje koristeći metodu *reshape* i standarizirali vrijednosti piksela slikovnih podataka.

Na kraju smo pretvorili slikovne podatke i odgovarajuće oznake *y*, *y_trains*, *y_val* i *y_test* u NumPy polja zbog kompatibilnosti s raznim bibliotekama i okvirima strojnog učenja.

7. Računanje točnosti

Ocjena točnosti u strojnom učenju je metrika procjene koja mjeri broj točnih predviđanja napravljenih od strane modela u odnosu na ukupni broj napravljenih predviđanja. Izračunali smo je tako što smo izračunali točnost svake klase i uzeli srednju vrijednost točnosti (“točnost točnosti”) jer testni skup nema jednak broj podataka za svaku klasu.

8. Implementacija neuronske mreže

Postupak prepoznavanja novčanica sastoji se od sljedećih koraka:

1. **Priprema skupa podataka** - potrebno je prikupiti skup novčanica i stvoriti bazu podataka. Novčanice moraju biti u digitalnom formatu kako bi se mogle koristiti u neuronskoj mreži. Svaka novčanica mora biti klasificirana prema svojoj vrijednosti.

2. **Treniranje neuronske mreže** - u ovom koraku se neuronska mreža uči prepoznavati novčanice. To se postiže iterativnim procesom u kojem se mreža šalje ulazne slike novčanica, a izlazne vrijednosti uspoređuju se s točnim vrijednostima novčanica. Nakon što se mreža pravilno nauči prepoznavati novčanice, spremna je za upotrebu u stvarnom svijetu.
Kako bi trenirali našu neuronsku mrežu, napravili smo petlju u modelu strojnog učenja pomoću Keras API-ja. Koristeći funkciju *fit()*, trenirali smo model na podacima koje daje generator *train_generator* za 100 epoha. Tijekom obuke, model se potvrđuje na temelju podataka koje daje *val_generator* generator. Argument *verbose* određuje opširnost izlaza tijekom obuke. Vrijednost 1 znači da će se trake napretka prikazati za svaku epohu. Argument povratnih poziva koristi se za određivanje povratnih poziva koji bi se trebali koristiti tijekom obuke. U ovom slučaju, *checkpoint_callback* je osiguran za spremanje težine modela nakon svake epohe.

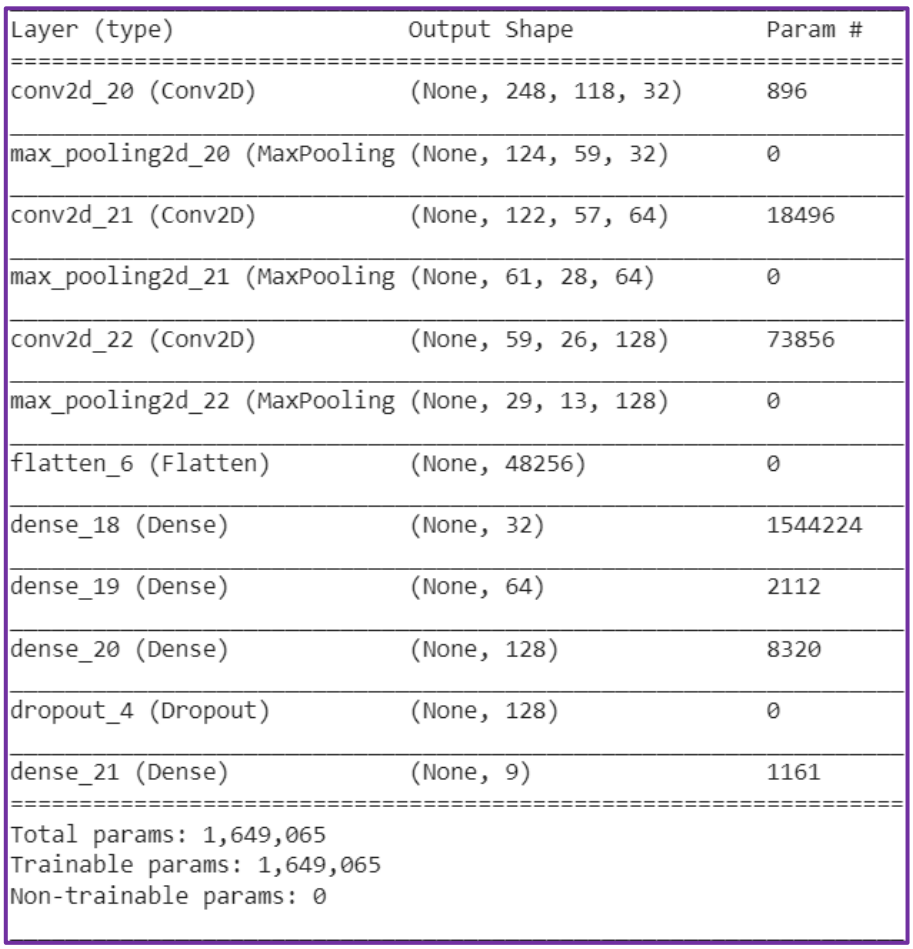
3. **Testiranje neuronske mreže** - posljednji korak u postupku prepoznavanja novčanica je testiranje neuronske mreže. Testiranje se provodi korištenjem skupa slika koje se nisu koristile u treniranju mreže. Ovim se postupkom provjerava točnost prepoznavanja novčanica.
Koristili smo funkciju *custom_evaluation* za procjenu izvedbe modela klasifikacije koristeći srednju ocjenu točnosti klase kao metriku procjene. Funkcijom *load_model* učitali smo spremljeni Keras model. Klasa *ImageDataGenerator* koristi se za povećanje podataka i pretprocesiranje slikovnih podataka u Kerasu. Generira serije slikovnih podataka s povećanjem podataka u stvarnom vremenu. Instanca *test_datagen* primjenjuje ponovno skaliranje na ulazne slike dijeljenjem svake vrijednosti piksela s 255.0. Metoda *flow_from_directory* klase *ImageDataGenerator* korištena je za generiranje serija slikovnih podataka iz direktorija. Zatim smo procijenili izvedbu modela na

testnom skupu na sljedeći način: generirali smo predviđanja za testni skup koristeći uvježbani model i *test_generator* te ih pohranili u varijablu *y_pred*. Metoda *predict_generator()* primjenjuje model na svaku skupinu podataka u testnom skupu i vraća predviđene vrijednosti. Dohvatili smo prave oznake skupa testova iz klasnog atributa *test_generator*-a i pohranili ih u *y_true*. Ovaj atribut sadrži cjelobrojne oznake klase koje odgovaraju svakom uzorku u testnom skupu. Na kraju smo provjerilli jesu li duljine *y_pred* i *y_true* jednake. Ova tvrdnja je važna jer bi broj predviđanja generiranih modelom trebao odgovarati broju pravih oznaka u testnom skupu.

8.1. Treniranje neuronske mreže

Kao vrstu neuronske mreže za prepoznavanje novčanica, zbog pogodnosti u radu s predikcijom fotografija, odabiremo konvolucijsku neuronsku mrežu.

Neuronska mreža bit će implementirana pomoću biblioteke *keras* koja uvelike olakšava proces kreiranja i rada s neuronskim mrežama. Sljedeća slika prikazuje model implementirane neuronske mreže:



Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 248, 118, 32)	896
max_pooling2d_20 (MaxPooling	(None, 124, 59, 32)	0
conv2d_21 (Conv2D)	(None, 122, 57, 64)	18496
max_pooling2d_21 (MaxPooling	(None, 61, 28, 64)	0
conv2d_22 (Conv2D)	(None, 59, 26, 128)	73856
max_pooling2d_22 (MaxPooling	(None, 29, 13, 128)	0
flatten_6 (Flatten)	(None, 48256)	0
dense_18 (Dense)	(None, 32)	1544224
dense_19 (Dense)	(None, 64)	2112
dense_20 (Dense)	(None, 128)	8320
dropout_4 (Dropout)	(None, 128)	0
dense_21 (Dense)	(None, 9)	1161

Total params: 1,649,065
Trainable params: 1,649,065
Non-trainable params: 0

Slika Model implementirane neuronske mreže

U procesu treniranja koristi se optimizator Adam i funkcija gubitka Categorical Cross Entropy.

8.2. Testiranje neuronske mreže

Testiranjem neuronske mreže dobivena je točnost 98.20% te je matrica konfuzije sljedeća:

Slika Konfuzijska matrica za neuronsku mrežu

9. KNN klasifikator

KNN (K-Nearest Neighbors) jedan je od jednostavnih i popularnih algoritama za klasifikaciju u strojnom učenju. Algoritam pronalazi K najbližih susjeda za ulazni podatak i zatim klasificira taj podatak prema većinskom razredu tih K susjeda. Učinkovitost KNN-a ovisi o veličini skupa podataka i broju značajki. KNN ne radi dobro na podacima s puno šuma i visokom dimenzionalnošću.

Model KNN (k najbližih susjeda) klasifikatora jednostavno se implementira korištenjem modula *sklearn.neighbors* (više o tome na https://scikit-learn.org/stable/modules/neighbors.html#classification) .

Implementirali su se klasifikatori KNN s parametrima k = 3, 5, 7, 9, 13 te su dobivene sljedeće točnosti klasifikacije nad podacima za validaciju:

Vrijednost k	**3**	**5**	**7**	**9**	**13**
Točnost	80.18 %	79.88 %	79.27 %	79.57 %	76.22 %

Iz dobivenih rezultata vidljivo je da je točnost najveća za k = 3. Točnost KNN klasifikatora (za k = 3) nad testnim podacima iznosi 68.47%. Sljedeća slika prikazuje konfuzijsku matricu KNN klasifikatora:

Slika Konfuzijska matrica za klasifikator 3-nn

10. SVC klasifikator

SVC (Support Vector Classification) je algoritam strojnog učenja za binarnu klasifikaciju (razdvajanje podataka u dvije klase) i multiklasnu klasifikaciju (razdvajanje podataka u više od dvije klase). Ovaj algoritam se temelji na konceptu podrživih vektora i traži hiperravninu koja najbolje razdvaja različite klase.

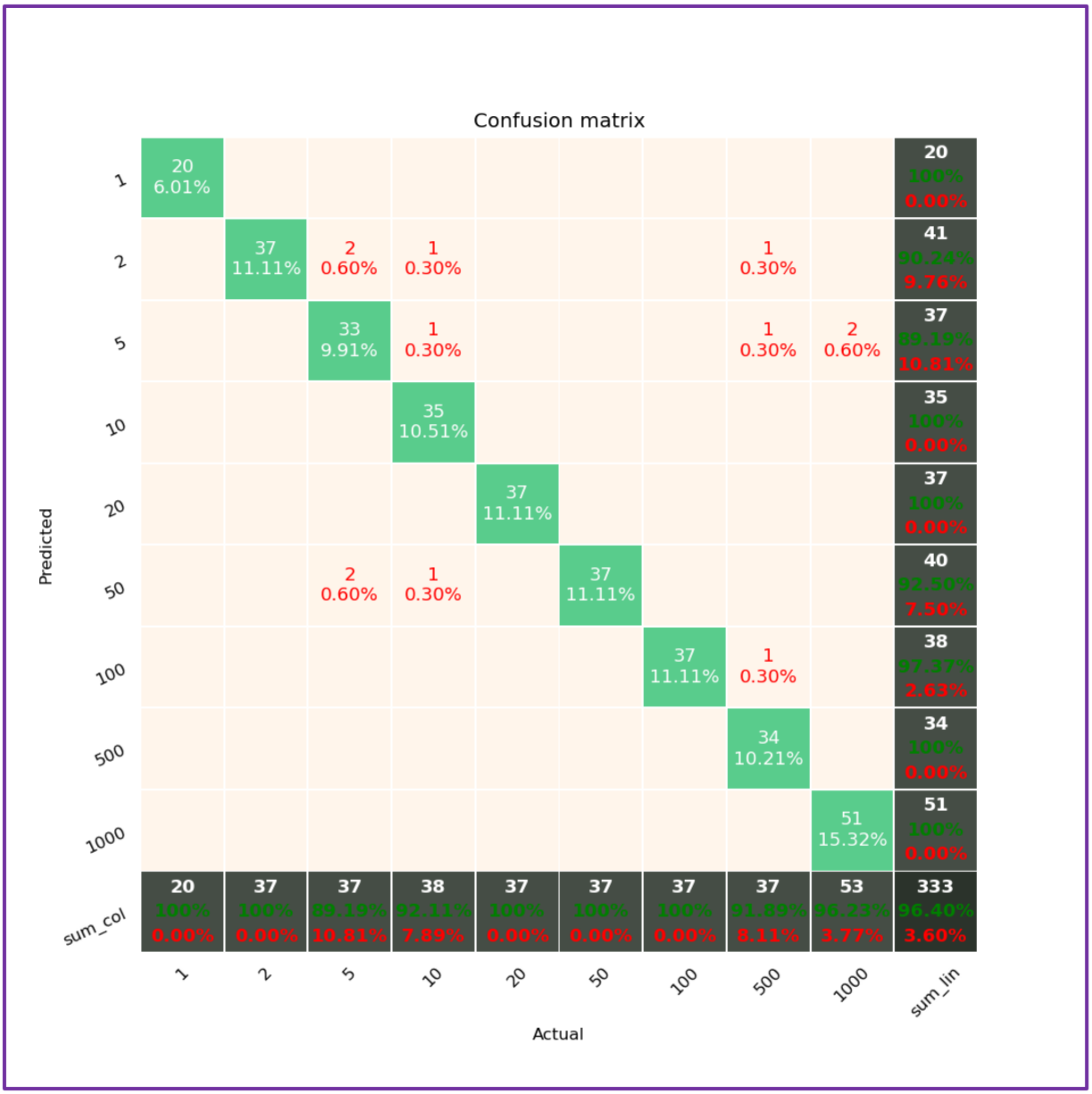
SVC klasifikator radi na sljedeći način:
1. Algoritam traži hiperravninu koja najbolje razdvaja dvije klase u prostoru značajki. Ta hiperravnina se naziva granica odluke.
2. Algoritam pronalazi podržavajuće vektore, odnosno primjere podataka koji su najbliži granici odluke. Ti primjeri podataka su bitni jer utječu na položaj granice odluke.
3. Kada dobijemo novi ulazni podatak, algoritam provjerava na kojoj strani granice odluke se nalazi taj podatak i tako ga klasificira u jednu od dvije klase.
SVC koristi funkciju izgleda koja optimizira granicu odluke. Najčešće se koriste RBF (radialna bazna funkcija) i linearna funkcija za pronalaženje granice odluke. Važno je napomenuti da je odabir odgovarajuće funkcije izgleda i parametara koji se koriste u funkciji važan za točnost i performanse modela.

Model SVC klasifikatora također se jednostavno implementira korištenjem modula *sklearn.svm.SVC* (više o tome na https://scikit-learn.org/stable/modules/svm.html#svm-classification) .

Implementirali su se SVC klasifikatori s jezgrama 'linear', 'poly', 'rbf', 'sigmoid' te su dobivene sljedeće točnosti klasifikacije nad podacima za validaciju:

Jezgra	**linear**	**poly**	**rbf**	**sigmoid**
Točnost	96.65 %	67.99 %	94.82 %	51.22 %

Klasifikator s najvećom točnošću je s jezgrom 'linear' i točnost nad testnim podacima iznosi 96.40%. Sljedeća slika prikazuje konfuzijsku matricu:



Slika Konfuzijska matrica za SVC klasifikator jezgre linear

11. Greške u konfuzijskim matricama

Konfuzijske matrice su odličan pristup za procjenu klasifikacijskog modela. Omogućuju točan uvid u to koliko je model ispravno klasificirao klase ovisno o unesenim podacima ili kako su klase pogrešno klasificirane. Svaki stupac konfuzijske matrice predstavlja instance stvarne klase, dok svaki redak predstavlja instance predviđene klase. Najčešće greške u konfuzijskim matricama su klasifikacija broja 2 kao 5 i obrnuto. Česta greška je i klasifikacija broja 5 kao 50, 10 kao 100 ili 1000, tj. da se neka nula zanemari ili propusti.

12. Zaključak

Testiranjem istog skupa podataka na različitim vrstama klasifikacije rezultiralo je različitim točnostima. Od implementiranih k-nn klasifikatora najveću točnost postigao je 3-nn i ona iznosi 68.47%. Nadlje, od SVC klasifikatora najveću ima onaj s jezgrom tipa *linear* koja iznosi 96.40%. Najveću točnost ima implementirana konvolucijska neuronska mreža koja ima točnost 98.20%.

Iako je neuronska mreža apstraktnija za implementaciju, ona se pokazala točnijom na problemu prepoznavanja novčanica. Točnost od 98.20% je velika, međutim, ako konkretna primjena algoritma zahtijeva još veću točnost, ona se može postići povećanjem broja podataka za treniranje i eventualnim prilagođavanjem modela.

Zaključak je da su neuronske mreže moćan alat i zasigurno područje koje će se i u budućnosti duboko proučavati i primjenjivati.

13. Literatura

- https://aws.amazon.com/what-is/neural-network/
- https://en.wikipedia.org/wiki/Convolutional_neural_network
- https://www.kaggle.com/
- https://www.kaggle.com/datasets/nsojib/bangla-money
- https://scikit-learn.org/stable/modules/neighbors.html#classification
- https://scikit-learn.org/stable/modules/svm.html#svm-classification