Klasifikacija bolesti lista kukuruza na digitalnim slikama korištenjem konvolucijskih neuronskih mreža

Galijan, Stjepan

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:835542

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-02-20**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Sveučilište Jurja Dobrile u Puli
Tehnički fakultet u Puli

STJEPAN GALIJAN

KLASIFIKACIJA BOLESTI LISTA KUKURUZA NA DIGITALNIM SLIKAMA KORIŠTENJEM KONVOLUCIJSKIH NEURONSKIH MREŽA

Diplomski rad

Pula, rujan 2024. godine

Sveučilište Jurja Dobrile u Puli
Tehnički fakultet u Puli

STJEPAN GALIJAN

KLASIFIKACIJA BOLESTI LISTA KUKURUZA NA DIGITALNIM SLIKAMA KORIŠTENJEM KONVOLUCIJSKIH NEURONSKIH MREŽA

Diplomski rad

JMBAG: 0303092236, redoviti student

Studijski smjer: Računarstvo

Predmet: Digitalna obrada i analiza slike

Znanstveno područje: Tehničke znanosti
Znanstveno polje: Računarstvo
Znanstvena grana: Umjetna inteligencija
Mentor: doc. dr. sc. Nikola Lopac

Pula, rujan 2024. godine

Doc. dr. sc. Nikola Lopac
(Ime i prezime nastavnika)

Digitalna obrada i analiza slike
(Predmet)

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet u Puli

ZADATAK TEME DIPLOMSKOG RADA

Pristupniku **MBS:**

Stjepan Galijan **0303092236**

Studentu Tehničkog fakulteta u Puli, izdaje se zadatak za diplomski rad – tema rada pod nazivom:

KLASIFIKACIJA BOLESTI LISTA KUKURUZA NA DIGITALNIM SLIKAMA KORIŠTENJEM KONVOLUCIJSKE NEURONSKE MREŽE

Sadržaj zadatka:

U radu je potrebno primijeniti metodu dubokog učenja zasnovanu na konvolucijskoj neuronskoj mreži za klasifikaciju digitalnih slika s prikazima bolesti lista kukuruza. Potrebno je opisati odabranu arhitekturu konvolucijske neuronske mreže, implementirati je, provesti proces učenja i testirati je na odabranom skupu slikovnih podataka. Potrebno je provesti analizu dobivenih rezultata korištenjem mjera učinkovitosti klasifikacije te provesti evaluaciju i diskusiju učinkovitosti odabrane metode u ovoj klasifikacijskoj primjeni.

Rad obraditi sukladno odredbama Pravilnika o završnom/diplomskom radu Sveučilišta u Puli.

Stjepan Galijan
(Ime i prezime studenta):

Redovni
(status izvanredni/redovni)

Diplomski sveučilišni studij Računarstvo
(studij)

Datum: 9. ožujka 2024.

Potpis nastavnika Nikola Lopac

Sadržaj

1. Uvod ... 1
2. Strojno učenje ... 3
3. Duboko učenje ... 4
4. Neuronske mreže ... 5
4.1. Povijest neuronskih mreža ... 6
4.2. Konvolucijske neuronske mreže ... 7
4.2.1. Konvolucijski sloj ... 8
4.2.2. Sloj sažimanja ... 10
4.2.3. Potpuno povezani sloj ... 10
4.3. Aktivacijske funkcije ... 11
4.3.1. Sigmoidna ... 11
4.3.2. Hiperbolička tangenta ... 11
4.3.3. ReLu ... 12
4.3.4. Softmax ... 13
5. Opis podataka i alata korištenih u radu ... 14
5.1. Bolesti kukuruza obrađene u radu ... 14
5.1.1. Siva pjegavost ... 14
5.1.2. Hrđa ... 15
5.1.3. Sjeverna plamenjača ... 16
5.2. Alati korišteni u izradi rada ... 16
5.2.1. Python ... 17
5.2.2. Tensorflow ... 17
5.2.3. Numpy ... 17
5.2.4. Matplotlib ... 18
6. Klasifikacija bolesti lista kukuruza na slikama ... 19
7. Rezultati i diskusija ... 28
8. Zaključak ... 34
Popis literature ... 35
Popis slika ... 37
Popis tablica ... 39
Sažetak ... 40
Abstract ... 41

1. Uvod

Kukuruz je jedan od najvažnijih usjeva u svijetu, te služi kao hrana za ljude, stoku i kao sirovina za razne industrijske proizvode. Međutim, usjev kukuruza je često izložen različitim bolestima koje mogu znatno smanjiti prinos i kvalitetu zrna. Otkrivanje i klasifikacija bolesti lista kukuruza ključni su za smanjenje ekonomskih gubitaka i osiguranje stabilnog prinosa, pogotovo ako se bolesti otkriju pravovremeno.

Razvojem tehnologije digitalne obrade slike i umjetne inteligencije, dolazi do novih i boljih mogućnosti za automatsko prepoznavanje i klasifikaciju bolesti na temelju digitalnih slika. Konvolucijske neuronske mreže (CNN) posebno su učinkovite u prepoznavanju obrazaca na slikama i stoga su idealne za zadatke poput klasifikacije biljnih bolesti.

U ovom radu istraživat će se metode dubokog učenja, posebno konvolucijske neuronske mreže, za klasifikaciju bolesti lista kukuruza. Cilj je razviti model koji će biti sposoban prepoznati i klasificirati najčešće bolesti lista kukuruza, kao što su siva pjegavost, hrđa i sjeverna plamenjača, a u radu će se opisivati strojno i duboko učenje, opisivat će se neuronske mreže, bolesti i alati potrebni za izradu ovog rada. Također, provodit će se implementacija modela, te analiza i evaluaciju dobivenih rezultata. Rad će uključivati prikupljanje relevantnih podataka, njihovu obradu i pripremu za treniranje modela, te samu izradu i testiranje konvolucijske neuronske mreže.

Očekuje se da će rezultati ovog istraživanja pružiti koristan alat za poljoprivrednike i istraživače, te doprinijeti unapređenju sustava za rano otkrivanje bolesti lista kukuruza. Cilj je također prikazati prednosti i izazove korištenja konvolucijskih neuronskih mreža u poljoprivrednom kontekstu te dati podlogu za buduća istraživanja u ovom području.

Rad započinje opisom strojnog učenja u drugom poglavlju, uključujući pojašnjenje vrsti strojnog učenja. Nakon toga treće poglavlje govori o dubokom učenju, njegovoj definiciji i kako ono funkcionira. Četvrto poglavlje govori o neuronskim mrežama, njenim vrstama i njihovim opisima. Također u navedenom poglavlju posebno se opisuju i konvolucijske neuronske mreže te njeni slojevi: konvolucijski sloj, sloj sažimanja i potpuno povezani sloj. U ovom se poglavlju može pronaći i povijest razvoja

neuronskih mreža, koja govori kako su nastale i kako su osmišljene. Potom u petom poglavlju opisuje se skup podataka koji je korišten u rad kao i bolesti lista kukuruza te korišteni programski alati. Šesto poglavlje govori o koracima koji su provedeni u izradi rada, odnosno implementaciji odgovarajućeg modela neuronske mreže, kao i o augmentaciji podataka, Adam optimizaciji i funkciji gubitka. U sedmom poglavlju prikazani su i analizirani dobiveni rezultati. Osmo poglavlje donosi glavne zaključke rada.

2. Strojno učenje

Strojno učenje (*engl. machine learning*) grana je umjetne inteligencije (*engl. artificial intelligence*) kojoj je u cilju stvaranje algoritama i modela koji pomažu računalima da nauče iz analiziranih podataka i poboljšavaju svoju izvedbu bez eksplicitnog programiranja. Jednostavnije rečeno, umjesto da osoba ručno unosi svaki korak, strojno učenje omogućuje računalima da sami prepoznaju obrasce u podacima i donose predviđanja ili odluke na temelju tih obrazaca. Proces strojnog učenja počinje prikupljanjem podataka, koji mogu biti u različitim oblicima, poput slika, teksta, brojeva ili zvukova. Ti se podaci zatim čiste, organiziraju i pripremaju za korištenje algoritmima strojnog učenja. Odabire se odgovarajući model, poput linearne regresije, stabla odlučivanja ili neuronske mreže, koji se trenira na pripremljenim podacima. Model prilagođava svoje parametre kako bi bolje prepoznao uzorke. Nakon što model završi s treniranje, započinje testiranje na zasebnim podacima kako bi se potvrdila njegova točnost i preciznost. Ako je model dovoljno točan, može se koristiti za predviđanje ili donošenje odluka o novim podacima. (Trask, 2019) (Géron, 2019) Postoje tri osnovne vrste strojnog učenja (Géron, 2019) (Trask, 2019) (Campesato, 2020):

- Nadzirano učenje: model uči iz označenih podataka kako bi predvidio izlaze za nove ulaze.
- Učenje bez nadzora: model identificira skrivene uzorke u neoznačenim podacima.
- Podržano učenje: Model je u interakciji sa svojom okolinom i uči iz povratnih informacija u obliku nagrada ili kazni.

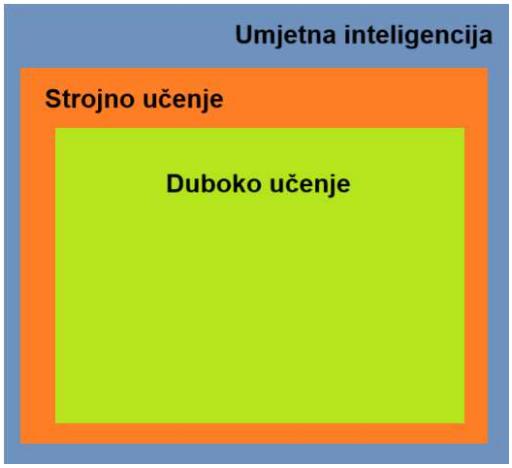
Primjena strojnog učenja pokriva različita područja, kao što su prepoznavanje slike i govora, obrada prirodnog jezika, medicinska dijagnostika, samovozeći automobili i mnoga druga područja.

3. Duboko učenje

Ključna grana strojnog učenja je duboko učenje, koje je nastalo sredinom 20. stoljeća. Modeli dubokog učenja izgrađeni su na perceptronima kao temeljima neuronskih mreža, obično koristeći velike ili opsežne skupove podataka. Ovi modeli također uključuju heuristiku i oslanjaju se na empirijske nalaze. (Campesato, 2020) (Trask, 2019)

Duboko učenje (*engl. deep learning*) je podskup strojnog učenja (primjer se može vidjeti na Slici 1 koja prikazuje podskupove umjetne inteligencije) koji uključuje neuronske mreže (*engl. neural networks*) s mnogo slojeva, poznate kao duboke neuronske mreže, za modeliranje složenih obrazaca i odnosa u podacima. Karakterizira ga njegova sposobnost automatskog učenja hijerarhijskih prikaza podataka kroz više slojeva apstrakcije, što ga čini posebno učinkovitim za zadatke koji uključuju nestrukturirane podatke kao što su slike, zvuk i tekst. Svaki sloj u dubokoj neuronskoj mreži transformira ulazne podatke u sve apstraktnije i složenije prikaze, omogućujući modelu da uhvati razne obrasce. (Campesato, 2020) (Trask, 2019)

Ovaj proces je olakšan naprednim tehnikama optimizacije i snažnim računalnim resursima, omogućujući modelima dubokog učenja postizanje najsuvremenijih performansi u različitim domenama, uključujući računalni vid, obradu prirodnog jezika i prepoznavanje govora. Kapacitet dubokog učenja za rukovanje velikim količinama podataka i njegova fleksibilnost u arhitekturi modela čine ga temeljnom tehnologijom u razvoju aplikacija umjetne inteligencije. (Campesato, 2020) (Trask, 2019)



Slika 1. Odnosi između umjetne inteligencije, strojnog učenja i dubokog učenja

Izvor: Izradio autor

4. Neuronske mreže

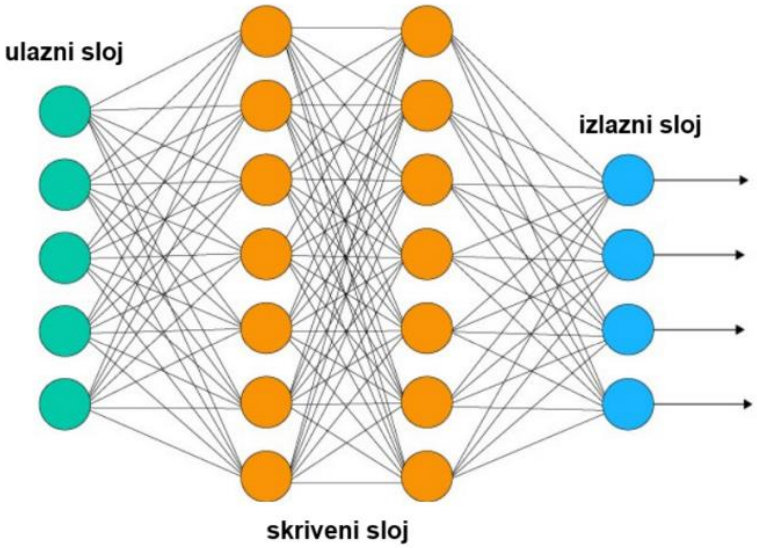
Neuronska mreža je inspirirana ljudskim mozgom, koji se sastoji od međusobno povezanih čvorova (neurona) koji obrađuju i prenose informacije, primjer se može vidjeti na Slici 2. Također ona je algoritam strojnog učenja. Neuroni su organizirani u ulazni i izlazni sloj te jedan ili više skrivenih slojeva koji se nalaze između njih. Ulazni sloj prima podatke te ih šalje u skrivene slojeve gdje se ti podatci dalje obrađuju. Na primjer, mreži je data slika kruga koja ima dimenzije 28 piksela x 28 piksela (28 x 28 = 784 piksela), na ulaznom sloju svaki se piksel dodjeljuje jednom neuronu u prvom redu. Neuroni u prvom redu su povezani sa neuronima u drugom redu uz pomoć kanala koji imaju dodijeljenu vrijednost ili težinu (*engl. weights*). Nakon toga ulazi se množe sa pripadajućim težinama, te se njihova suma šalje dalje na sljedeće neurone gdje joj se pridoda (zbroji) i vrijednost „Bias“ (vrijednost koja se pridodaje neuronu, te omogućuje aktivacijskoj funkciji pomicanje lijevo ili desno kako bi bolje odgovarala podatcima). Potom ta vrijednost prolazi kroz aktivacijsku funkciju, čiji je zadatak odrediti hoće li određeni neuron biti aktiviran ili ne (primjer neuronske mreže se može vidjeti na Slici 2). Aktivirani neuroni zatim šalju signale neuronima u sljedećem redu putem kanala, i na taj način se vrijednosti šire kroz mrežu. Na kraju, neuron s najvećom vrijednosti određuje i donosi rezultat. (What is a Neural Network?) (Types of Neural Networks and Definition of Neural Network, 2024)

Postoji više vrsta neuronskih mreža i svaka odgovara različitim tipovima zadataka (What is a neural network?) (8 Common Types of Neural Networks, 2024) (Types of Neural Networks and Definition of Neural Network, 2024):

- Feedforward Neural Network (FNN) – najjednostavniji oblik neuronske mreže, a služi u zadatcima prepoznavanja uzoraka, regresije i klasifikacije
- Convolutional Neural Network (CNN) – ova mreža je dizajnirana za obradu podataka koji se mogu prikazati u mreži, kao što su slike. Njena upotreba se može naći u klasifikaciji slika, prepoznavanju i pronalaženju objekata na slikama i videima.
- Recurrent Neural Network (RNN) – mreža s ugrađenim petljama (neuron na ulaz dodatno dobiva i svoj izlaz), služi za prepoznavanje govora, obradu prirodnog jezika (NLP)

- Long Short-Term Memory Networks (LSTM) – generiranje teksta, sinteza govora
- Gated Recurrent Unit (GRU) – koristi se u zadatcima kojima je potrebno učenje sekvencijalnih podataka
- Autoencoder – uklanjanje šuma, kompresija podataka, otkrivanje anomalija
- Generative Adversarial Network (GAN) – generiranje slika i videa
- Radial Basis Function Network (RBFN) – kontrola sustava, klasifikacija
- Self-Organizing Maps (SOM) – vizualizacija podataka, otkrivanje anomalija
- Transformer Network– NLP obrada, prijevod i sažimanje teksta
- Capsule Network (CapsNet) – prepoznavanje slika i detekcija objekata

Postoje razne primjene neuronskih mreža, a neke od njih su prepoznavanje i generiranje slika i videa, u medicini, obrada prirodnog jezika, klasifikacija , generiranje i obrada teksta. (What is a Neural Network?) (Types of Neural Networks and Definition of Neural Network, 2024)



Slika 2. Osnovni prikaz slojeva duboke neuronske mreže

Izvor: Izradio autor

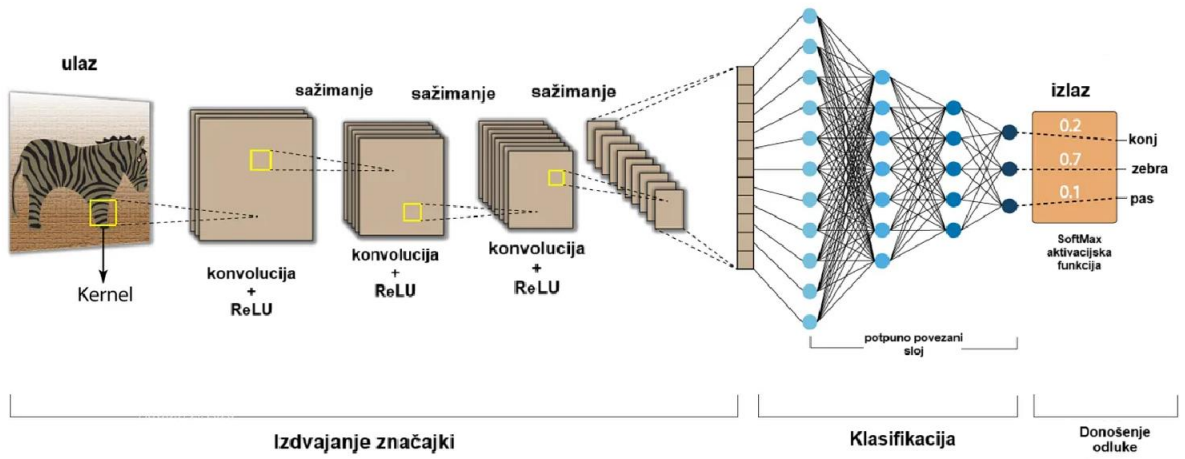
4.1. Povijest neuronskih mreža

Ideja neuronskih mreža je predstavljena 1943. godine od strane Warren McCullocha i Walter Pittsa. Oni su pokušali objasniti kako neuroni u mozgu zapravo rade na način

da su koristili elektroničke sklopove, jer su smatrali da se na ljudski mozak može gledati kao na računalo, no prvu neuronsku mrežu koja se može trenirati je osmislio Frank Rosenblatt, 1957. godine, pod nazivom perceptron. Perceptron je mogao naučiti težine od ulaznih podataka i nije imao skrivene slojeve, za razliku od današnjih modela neuronskih mreža. 1982. godine John Hopfield je razvio neuronske mreže s povratnim vezama koje su dobile naziv Hopfieldove mreže te pobudile interes u području neuronskih mreža. Ubrzo nakon Hopfieldove mreže, Terry Sejniwski i Geoffrey Hinton, 1985. razvijaju Boltzmannov stroj (stohastički stroj s povratnim vezama). Osamdesetih godina se također razvijaju neuronske mreže s više slojeva i imaju mogućnost rješavanja puno kompleksnijih zadataka. Takve neuronske mreže se nazivaju višeslojni perceptron (*engl. multi-layer perceptron*, MLP). Nakon stanke zbog ograničenja računalne snage, 1990-ih Vladimir Vapnik razvija stroj potpornih vektora (*engl. support vector machine*, SVM), što dovodi do evolucije strojnog učenja i neuronskih mreža, jer je on dobra zamjena za zadatke klasifikacije. Hochreiter i Schmidhuber su 1997. godine napravili dugu kratkoročnu memoriju (*engl. Long Short-Term Memory*, LSTM), koja rješavaja problem nestajućeg gradijenta u ponavljajućim neuronskim mrežama. Nakon svega u 1990-im godinama, ponovno dolazi do zastoja razvoja i istraživanja sve do 2006. godine kada dolazi novi naziv „Deep Learning“ ili duboko učenje. (Özer, 2023) (A Concise History of Neural Networks, 2016)

4.2. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (*engl. Convolutional Neural Networks*, CNN) klasa su modela dubokog učenja posebno dizajniranih za obradu i analizu struktura podataka nalik mreži, najčešće slika (Slika 3). Posebno su dobri u prepoznavanju obrazaca, što ih čini vrlo sposobnima za rješavanje zadataka računalnog vida. Primjena CNN-ova se također može pronaći u drugim raznim područjima, kao što su: klasifikacija slika (klasifikacija bolesti na raznim medicinskim slikama, prepoznavanje bolesti lista kukuruza i sl.), prepoznavanje lica (koriste se u autentifikacijskim i sigurnosnim sustavima), detekcija objekata (na primjer detekcija pješaka u autonomnim vozilima), segmentacija slika (klasificira se svaki piksel na slici kako bi se pronašle određene regije na slici) (Sewak, Karim i Pujari, 2018) (Géron, 2019).



Slika 3. Tipična konvolucijska neuronska mreža

Izvor: Autor modificirao prema: Naebi i Feng, 2023

CNN-ovi se sastoje od više slojeva koji pretvaraju ulazne podatke u izlazna predviđanja i rezultate kroz razne konvolucijske operacije, sažimanje (*engl. pooling*) i potpuno povezane slojeve. Za izradu konvolucijskih neuronskih mreža potrebna su tri glavna sloja (Sewak, Karim i Pujari, 2018) (Géron, 2019):

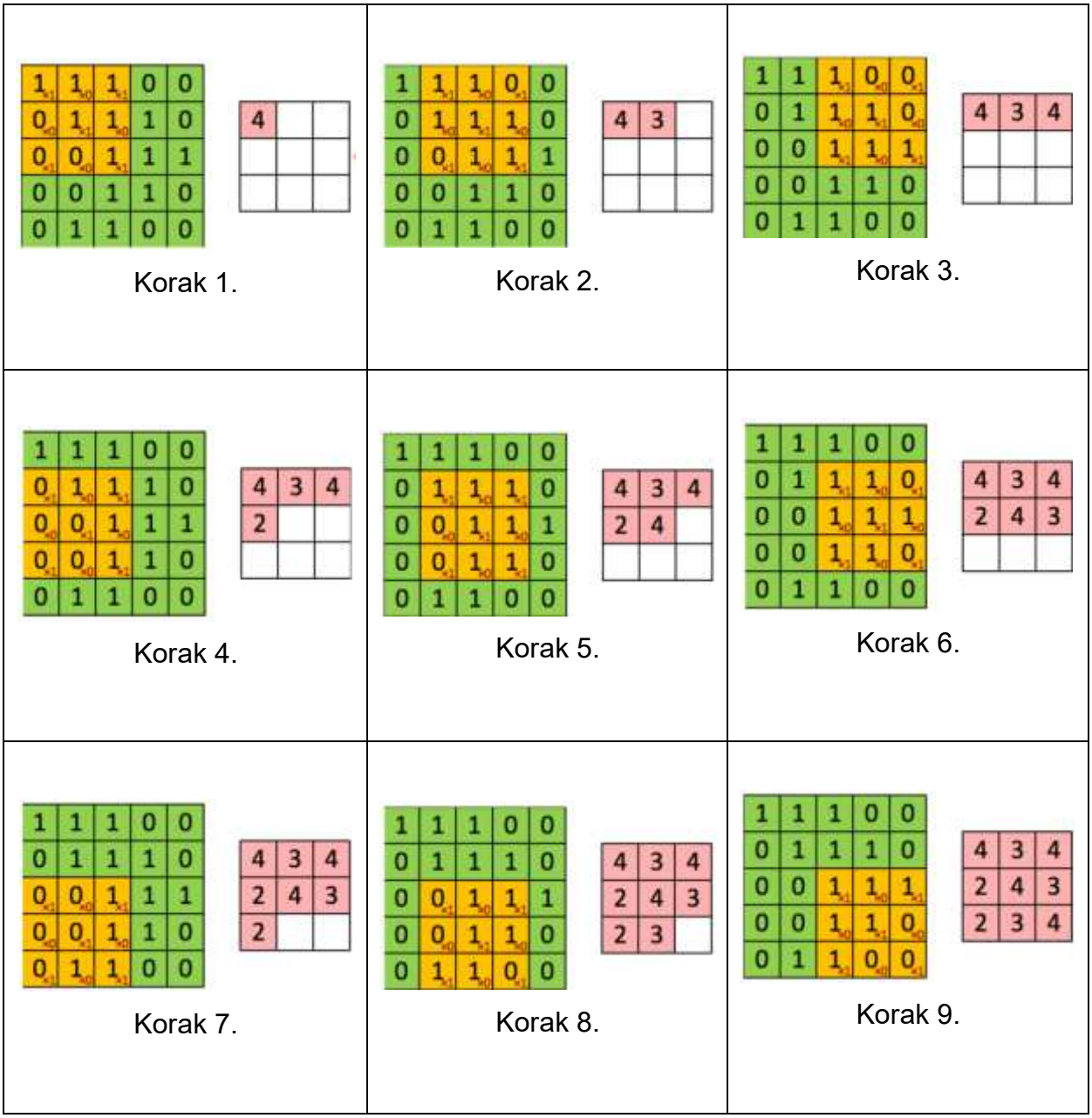
- Konvolucijski sloj (*engl. Convolutional layer*)
- Sloj sažimanja (*engl. Pooling layer*)
- Potpuno povezani sloj (*engl. Fully connected layer*)

4.2.1. Konvolucijski sloj

Konvolucijski sloj temeljna je komponenta konvolucijske neuronske mreže, a obično se koristi u dubokom učenju za zadatke koji uključuju prepoznavanje, klasifikaciju i obradu slike i videa. Ovaj sloj sadrži parametre koji se sastoje od skupova filtera koje je moguće naučiti, potom se ti filteri stavljaju preko ulaznih podataka te se množe (npr. gornji lijevi ulazni podatak se množi s gornjim lijevim brojem u filteru), primjer se može vidjeti u Tablici 1. Ovim procesom se dobiva dvodimenzionalna aktivacijska tablica koje se slaže i tako se dobiva izlazni produkt. Kao što je već spomenuto filteri su male matrice (npr. 3x3 ili 5x5) koje imaju razne parametre i koje su dizajnirane da prepoznaju različite rubove, teksture ili kompliciranije uzorke (Géron, 2019) (Sewak, Karim i Pujari, 2018). Ti parametri su:

- Korak (*engl. stride*) – koracima se određuje za koliko će se filter pomicati preko ulazne matrice. Jedan korak znači da se filter pomiče jedan po jedan piksel.

- Dubina (*engl. depth*) – dubina sloja
- Dopuna/ispuna nulama (*engl. zero-padding*) – dodavanje nula kako bi se mogla kontrolirati prostorna veličina izlazne mape

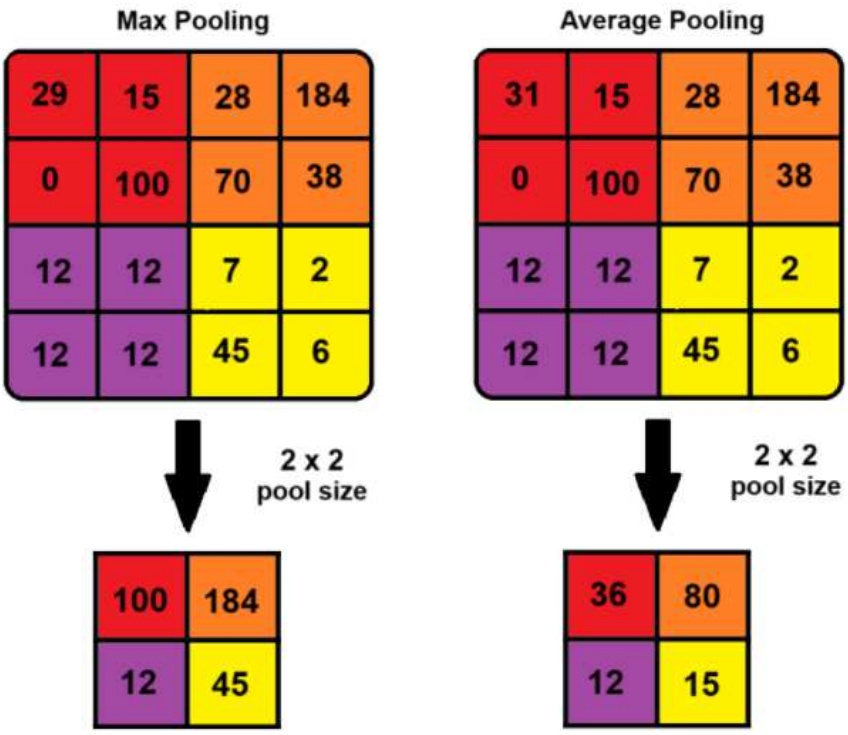


Tablica 1. Prikaz koraka u konvolucijskom sloju

Izvor: Autor modificirao prema: CPSC 440: Machine Learning, 2022

4.2.2. Sloj sažimanja

Sloj sažimanja je komponenta konvolucijskih neuronskih mreža, čija je primarna svrha smanjiti prostorne dimenzije ulaznih mapa kako bi se smanjilo opterećenje na računalo, upotreba memorije i broj parametara. Najčešće se postavlja između konvolucijskih slojeva, a najčešće vrste udruživanja su maksimalno (*engl. max*), prosječno (avg, *engl. average*) i globalno (*engl. global*). Sažimanje se najčešće radi unutar malog prozora, npr. 2x2, dok mu je stride ili broj koraka za koje se pomiče jednak veličini prozora, što znači da se pomiče za 2 mjesta kako ne bi došlo do preklapanja (Campesato, 2020) (Géron, 2019). Primjer kako radi sloj sažimanja se može vidjeti na Slici 4.



Slika 4. Prikaz funkcija sloja sažimanja, Max Pooling i Average Pooling

Izvor: Autor modificirao prema: How, 2023

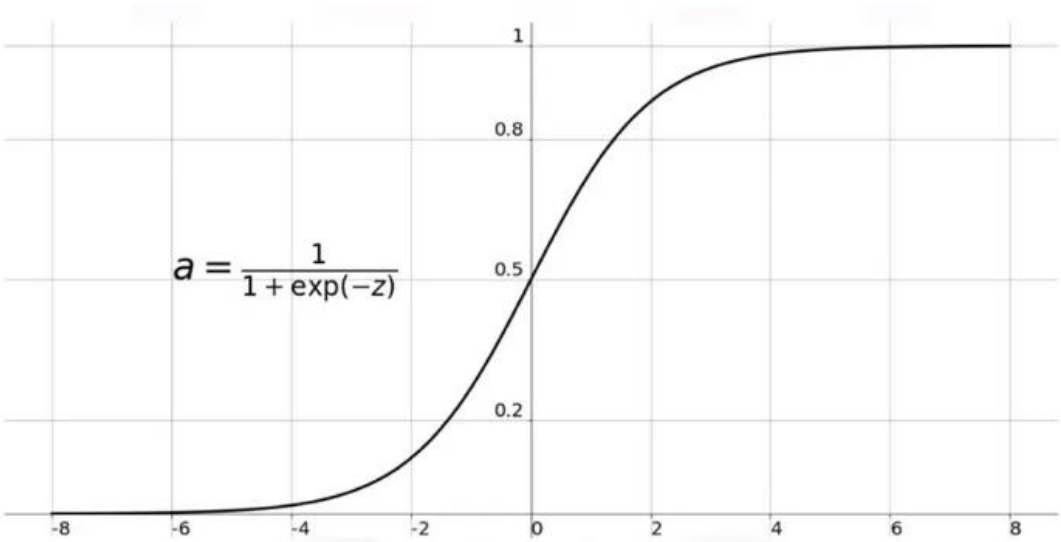
4.2.3. Potpuno povezani sloj

Dobio je naziv jer je svaki neuron u ovom sloju povezan sa svakim neuronom u prethodnom sloju.

4.3. Aktivacijske funkcije

4.3.1. Sigmoidna

Sigmoidna funkcija je krivulja u obliku slova S koja uzima bilo koju vrijednost između nula i jedan, te je vrlo korisna i prikladna za korištenje na izlaznom sloju za binarnu klasifikaciju. Ona je povijesno najvažnija aktivacijska funkcija, a modeli koji su je najviše koristili su modeli koji predviđaju neke vjerojatnosti (Sewak, Karim i Pujari, 2018) (Campesato, 2020). Primjer sigmoidne funkcije i njena formula se mogu vidjeti na Slici 5.

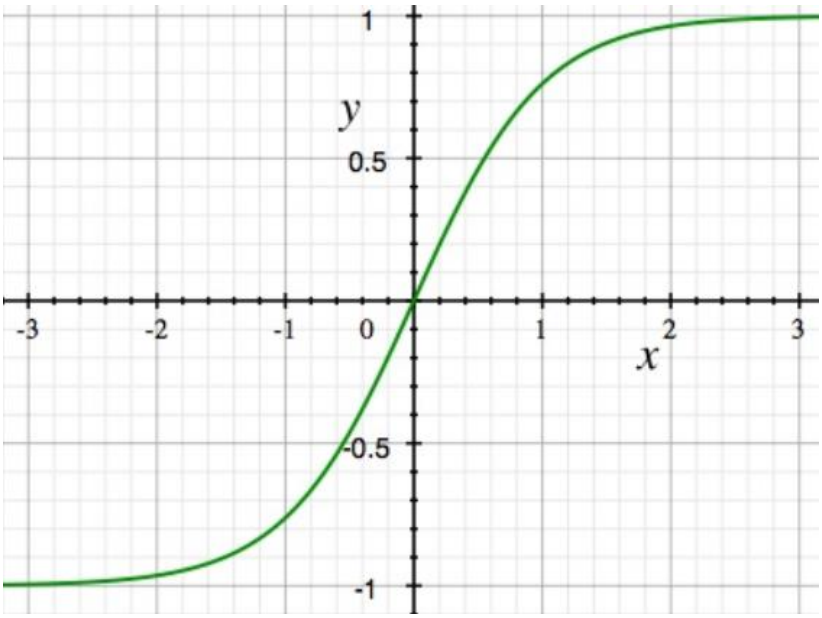


Slika 5. Graf sigmoidne aktivacijske funkcije

Izvor: Toprak, 2020

4.3.2. Hiperbolička tangenta

Hiperbolička tangenta je funkcija slična sigmoidnoj funkciji samo što se prostire između -1 i 1 umjesto 0 do 1 kao što to radi sigmoidna funkcija. Često se koristi u skrivenim slojevima te je centrirana oko nule (Campesato, 2020) (Sewak, Karim i Pujari, 2018). Ova funkcija može imati problema kod nestajanja gradijenata i njen primjer se može vidjeti na Slici 6.

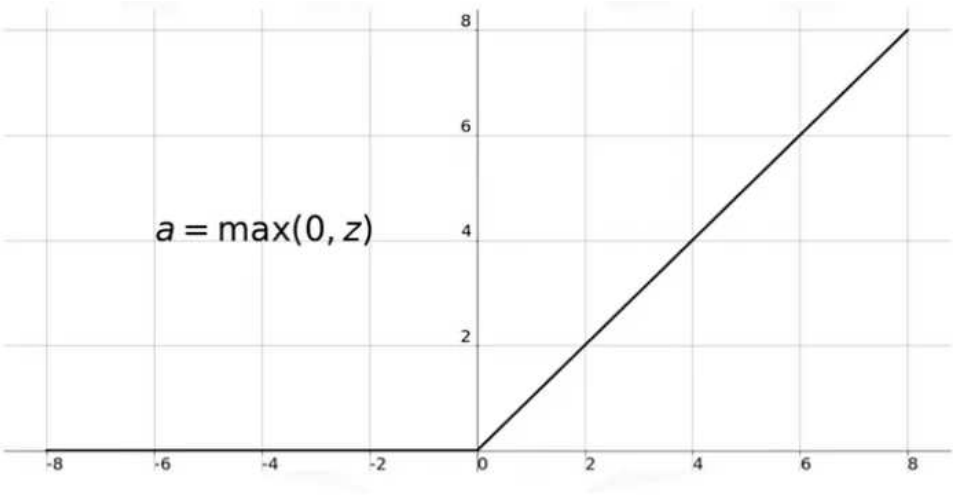


Slika 6. Graf aktivacijske funkcije hiperboličke tangente

Izvor: Tanh Activation Function

4.3.3. ReLu

ReLu ili Rectified Linear Unit (primjer se može vidjeti na Slici 7) je linearna funkcija gdje se ulaz ispravlja ako je manji ili jednak od 0 tako da se za svaki x koji je manji ili jednak 0, daje vrijednost nula na izlazu dok za svaki x koji je veći od 0, na izlazu se daje taj isti x i time se nadvladava problem nestajućeg gradijenta. Postoje još dvije vrste ReLu aktivacijske funkcije, a to su Leaky ReLu i ELU (Exponential Linear Unit) (Campesato, 2020) (Sewak, Karim i Pujari, 2018).

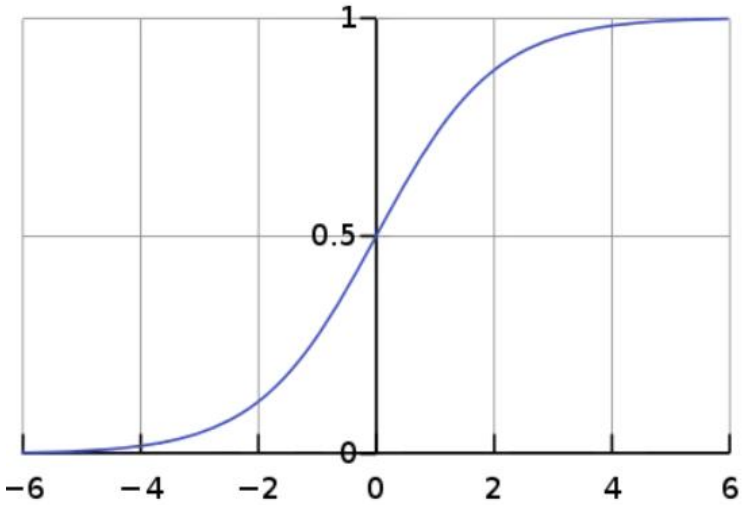


Slika 7. Graf ReLu aktivacijske funkcije

Izvor: Toprak, 2020

4.3.4. Softmax

Softmax funkcija se koristi na izlaznom sloju neuronske mreže kod problema sa klasifikacijom koja ima više klasa (*engl. multi-class classification*), a može se vidjeti na Slici 8. Radi na način da zaprima ulazne vrijednosti koje mogu biti nula, veće od 1, pozitivne i negativne ali ih ona svejedno pretvara u vrijednosti između 0 i 1 kako bi se na njih moglo gledati kao na vjerojatnosti (Campesato, 2020).



Slika 8. Graf softmax aktivacijske funkcije

Izvor: Softmax vs LogSoftmax, 2021

5. Opis podataka i alata korištenih u radu

Podatci nad kojima će se provoditi testiranja i analiza su uzeti s web stranice kaggle, s poveznice https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset, a skup podataka se zove „PlantVillage Dataset“ (PlantVillage Dataset). Kaggle je online zajednica i platforma koju koriste ljudi, a posebno znanstvenici koji se bave podatcima ili strojnim učenjem. Na kaggle-u se mogu pronaći razni skupovi podataka koji korisnicima mogu pomoći s njihovim kodom, aplikacijama, poslom i slično. Također, kaggle daje opciju kolaboracije gdje korisnici mogu unaprijediti svoje znanje i vještine.

Odabrani skup podataka sadrži slike lišća oboljelih biljaka, kao što su jabuka, kukuruz, grožđe, krumpir i druge, ali za izradu ovog rada korištene su slike sa bolestima lista kukuruza. U skupu podataka nalaze se slike zdravog lista kukuruza te oboljelih listova kukuruza od strane: sive pjegavosti, hrđe i sjeverne plamenjače. Sve slike su istih dimenzija: 256 x 256 piksela. Također slike su u JPG formatu dok je model boja RGB. U korištenom skupu podataka nalaze se 3852 slike.

5.1. Bolesti kukuruza obrađene u radu

5.1.1. Siva pjegavost

Siva pjegavost lista kukuruza je gljivična bolest uzrokovana gljivicama roda Ceracospora te se može vidjeti na Slici 9. Prepoznaje se po malim kružnim žutosmeđim mrljama s tamnijim rubovima na lišću. Što više bolest napreduje to se više mali krugovi spajaju i uzrokuju još veća oštećenja te smanjuju fotosintetske sposobnosti kukuruza, što dovodi do smanjenja prinosa usjeva. Neki od načina sprječavanja ove bolesti su korištenje fungicida, sadnja otpornih hibrida, sadnja u redovima. (Maize Diseases: A Guide for Field Identification, 2004)

Slika 9. Siva pjegavost lista kukuruza

Izvor: Uzorak iz PlantVillage Dataset

5.1.2. Hrđa

Karakteriziraju je nizovi izduženih ili okruglih kvrga kao što se može vidjeti na Slici 10. Hrđu je teško uočiti u ranim fazama jer su kvržice jako male, a dok se otkriju postoji mogućnost da te kvrge s vremenom puknu, čime šire spore dalje po drugom lišću. Kvrge su inače crvenkasto smeđe boje te se mogu naći i s gornje i donje strane lista. Hrđa se najčešće pojavljuje u periodima s visokim temperaturama i obilnim oborinama, koji također mogu dovesti i do raznih drugih oboljenja. Sprječavanje nastanka hrđe se odvija na slične načine kao i za sivu pjegavost lista, a to su pravilno prorjeđivanje, sadnja otpornijih vrsta kukuruza, upotreba fungicida te držanje polja provjetrenim. (Maize Diseases: A Guide for Field Identification, 2004)

Slika 10. Hrđa lista kukuruza

Izvor: Uzorak iz PlantVillage Dataset

5.1.3. Sjeverna plamenjača

Sjeverna plamenjača je gljivična bolest, a karakteriziraju je duge, sivo zelene ili bež, lezije koje nastaju paralelno s venama lista (Slika 11). Ova bolest često onemogućuje fotosintezu jer lezije rastu, te se spajaju i tako „onesposobljuju“ cijeli list. Ova bolest najčešće kreće od donjih listova prema gore, a spriječiti se može uz pomoć fungicida, pravljenjem plodoreda (izmjena biljaka koje se uzgajaju svake godine na određenom polju), također mogu se koristiti otporni hibridi. (Maize Diseases: A Guide for Field Identification, 2004)

Slika 11. Sjeverna plamenjača na listu kukuruza

Izvor: Uzorak iz PlantVillage Dataset

5.2. Alati korišteni u izradi rada

Za izradu ovog diplomskog rada korišten je programski jezik Python dok se na Slici 12 može vidjeti popis Python biblioteka koje su uz njega korištene te koje će biti pobliže objašnjene u nastavku teksta. Korištene biblioteke su Tensorflow, Numpy i Matplotlib.

```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Resizing, Rescaling
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import RandomFlip, RandomRotation
from tensorflow.keras.models import Sequential
import numpy as np
```

Slika 12. Popis Python biblioteka korištenih u radu

Izvor: Izradio autor

5.2.1. Python

Python je programski jezik koji je osmišljen 1991. godine od strane Guida van Rossuma, a može se koristiti u razne svrhe, kao što su podatkovna znanost (*engl. data science*), strojno učenje, web aplikacije, matematika, fizika, automatizacija, testiranje software. Objektno je orijentiran i interpretiran dok ga njegova jednostavna sintaksa čini jednim od najpopularnijih programskih jezika za brzi razvoj aplikacija, pisanje skripti i slično. Python podržava razne okvire (*engl. frameworks*) i biblioteke (*engl. libraries*) koje mu omogućavaju da što lakše obavlja prije navedene zadatke. (Python Introduction, 2024) (The Python Tutorial, 2024)

5.2.2. Tensorflow

Tensorflow je okvir za strojno učenje otvorenog koda poznat po pokretanju dubokih neuronskih mreža kodom visoke razine. Razvio ga je Googleov Brain tim za i prvi put je objavljen 2015. godine. U svojoj srži to je samo biblioteka za programiranje s linearnom algebrom i statistikom. Riječ „tensor“ opisuje multilinearni odnos između skupova algebarskih objekata unutar vektorskog prostora koji se zove višedimenzionalni niz. Ono što ga čini posebnim je zbirka API-ja za obradu podataka, vizualizaciju, evaluaciju modela i drugo. Koristi se u medicini za otkrivanje objekata i slika magnetskom rezonancom, za preporuke, obradu prirodnog jezika (*engl. Natural Language Processing*, NLP), samovozeće automobile i mnoge druge zadatke. (Introduction to TensorFlow) (Vaishya, 2023)

5.2.3. Numpy

Numpy je python biblioteka za rad sa nizovima, a napravio ju je Travis Oliphant, 2005. godine. Biblioteka je otvorenog koda i koristi se za linearnu algebru, statističku analizu, razne matematičke funkcije i slično, a napisana je uz pomoć Pythona i C. Numpy je napravljen za rad s Fourierovom transformacijom, slučajnim brojevima, N-dimenzionalnim nizovima i linearnom algebrom. Najviše se koristi u područjima kao što su podatkovna znanost, fizika, matematika, strojno učenje. (Vaishya, 2023) (NumPy documentation, 2024)

5.2.4. Matplotlib

Matplotlib je biblioteka koja služi za pravljenje interaktivnih vizualizacija podataka u Pythonu. S pomoću nje se mogu stvarati razni statički i dinamički grafikoni kao što su stupčasti grafovi, dijagrami, histogrami, kružni grafovi i mnogi drugi. Otvorenog je koda što joj daje veliku popularnost te se koristi u mnogim projektima. Osmislio ju je John D. Haunter, a napisana je uz pomoć programskih jezika C, Objective-C i Javascripta. (Matplotlib Tutorial, 2024) (Using Matplotlib, 2024) (Vaishya, 2023)

6. Klasifikacija bolesti lista kukuruza na slikama

Kukuruz je jedan od najvažnijih usjeva u svijetu te služi kao hrana za ljude, stoku i razne druge industrijske proizvode. Sklon je raznim bolestima koje mogu utjecati na kvalitetu i na doprinos pa rano otkrivanje istih može dovesti do sprječavanja nastanka štete. Ranim pronalaskom bolesti, poljoprivrednici mogu zaražene listove odstraniti, koristiti lijekove ili fungicide kako bi spriječili daljnje širenje. Također, mogu početi koristiti neke nove načine sadnje, navodnjavanja i slično jer se puno bolesti pojavljuje u kombinaciji vlage i visokih temperatura. Sprječavanje bolesti smanjilo bi doprinos i lošiju kvalitetu kukuruza, ali spriječilo bi i velike novčane gubitke, one koji su već uloženi i one koje bi osoba dobila prilikom prodaje ili daljnje investicije.

Kod klasifikacije bolesti lista kukuruza postoje i neki izazovi kao što su slike loše kvalitete (slike slikane u polju bi mogle lošije kvalitete zbog lošijih kamera ili različitih kutova slikanja ili osvjetljenja), također puno bolesti u nekim fazama razvoja imaju slične ili iste simptome. Rješenje za takve probleme se može naći u klasifikaciji bolesti lista kukuruza uz pomoć korištenja konvolucijskih neuronskih mreža, zbog njihove visoke učinkovitosti prilikom klasifikacije slika.

Prvi korak za izradu rada je prikupljanje podataka, ili u ovom slučaju slike zdravih listova kukuruza te zaraženih od strane sive pjegavosti, hrđe i sjeverne plamenjače. Prikupljanje se može vršiti na više načina dok su neki od njih korištenje već gotovih podataka sa web stranice kao što je kaggle, korištenje drugih alata koji će pretraživati Internet i sakupljati slike zaraženih listova kukuruza (web-scrape) ili imati tim fotografa i stručnjaka koji uzimaju fotografije raznih bolesti, otkrivaju koje su to bolesti i zatim ih tako spremaju u skup podataka.

Drugi korak izrade rada je priprema podataka (slika) koji će se obrađivati. Kao što je već prije navedeno, skup podataka se sastoji od 3852 slike, a od toga 1162 slike otpada na prikaz zdravog lista kukuruza, 1192 prikazuju hrđu lista kukuruza, njih 985 prikazuje bolest sjeverne plamenjače i njih 513 prikazuju bolest sive pjegavosti.

Na Slici 13 mogu se vidjeti postavljeni parametri. Parametar IMAGE_SIZE ima vrijednost 256 jer slike u skupu podataka imaju dimenzije, odnosno visinu i širinu,

jednaku 256 piksela. Također može se vidjeti parametar BATCH_SIZE koji ima vrijednost 32 što znači da će se u treniranju koristiti grupe od 32 slike odjednom. Parametar CHANNELS govori da su u pitanju slike RGB formata, a parametar EPOCHS govori da će biti 15 epoha.

```
IMAGE_SIZE = 256
BATCH_SIZE = 32
CHANNELS = 3
EPOCHS = 15
```

Slika 13. Postavljanje parametara

Izvor: Izradio autor

Slika 14 prikazuje unos svih slika iz skupa podataka koji dolaze iz mape plantvillage dataset te ih potom miješa iz razloga da model ne bi naučio kojim redoslijedom mu slike dolaze, jer bi to dovelo do lošeg treniranja modela i potencijalnih krivih rezultata. Na taj način svaki put dobije novi set slika na kojemu može trenirati. Na slici se također može vidjeti da su korišteni i parametri sa prijašnje slike kako bi postavili dimenzije slika i u kolikim grupama će slike dolaziti.

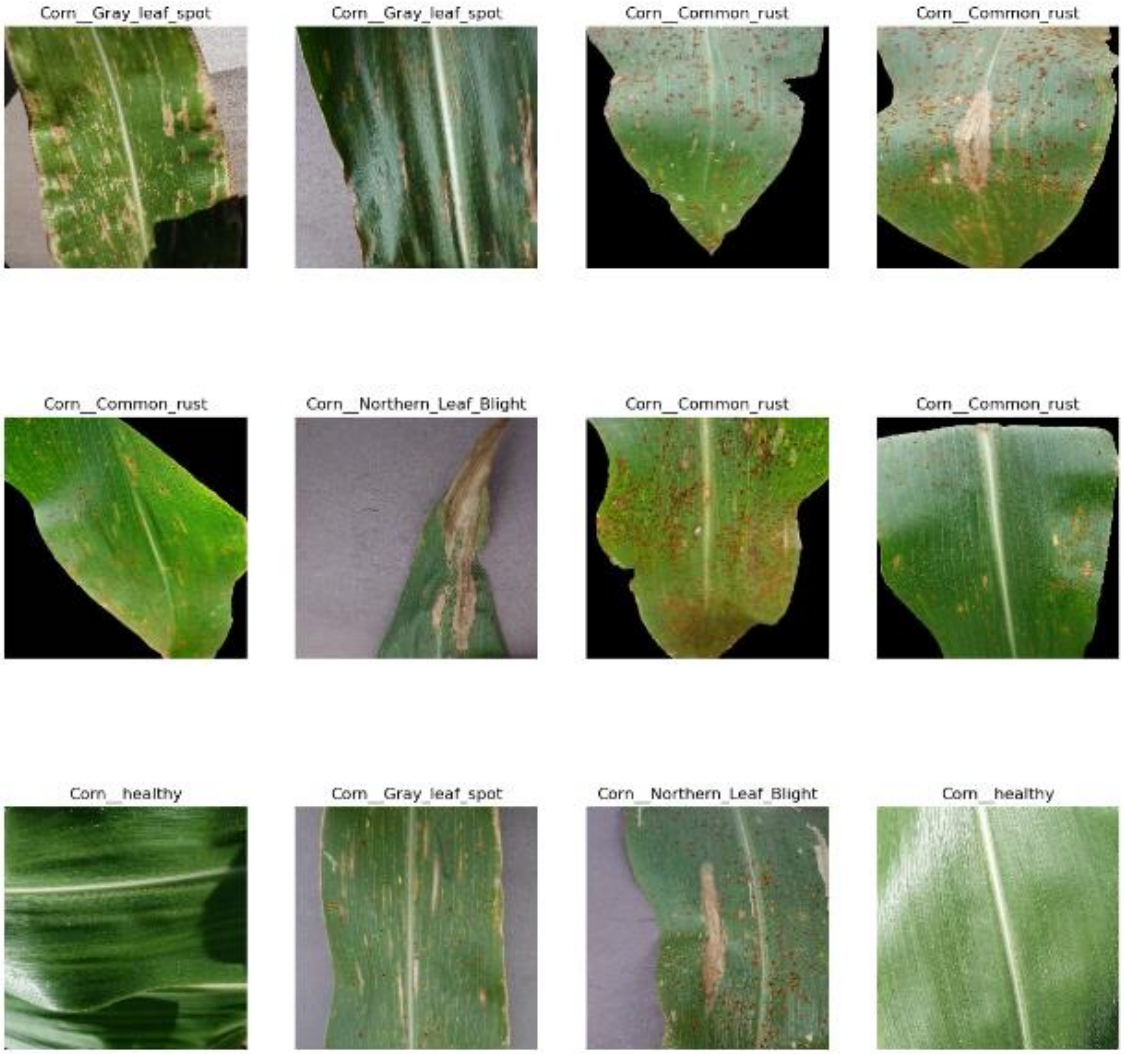
```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "plantvillage dataset",
    shuffle = True,
    image_size = (IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE
)

Found 3852 files belonging to 4 classes.
```

Slika 14. Unos podataka

Izvor: Izradio autor

Nakon unosa podataka, na Slici 15 se može vidjeti kod uz pomoć kojeg su slike iz skupa podataka prezentirane u Jupyter Notebooku, te imena iznad svake slike za koju su vezane.

```
plt.figure(figsize=(15, 15))
for image_batch, label_batch in dataset.take(1):
    for i in range(12):
        plt.subplot(3,4,i+1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[label_batch[i]])
        plt.axis("off")
```



Slika 15. Prikaz slika zdravih i bolesnih listova sa njihovim pripadajućim imenima

Izvor: Izradio autor

Sljedeći je korak raspodijeliti skup podataka u tri skupa, treniranje, validacija i testiranje. Skup za treniranje će koristiti 80% ili 3082 slike dok će skupovi za validaciju i testiranje svaki imati po 10% ili 385 slika. Na Slici 16 se mogu vidjeti ta tri gotova skupa koji imaju metode cache(), shuffle() i prefetch(). Metoda cache sprema podatke u priručnu memoriju što omogućava modelu da čita te podatke iz memorije, a ne sa diska čime se ubrzava treniranje. Shuffle metoda kao što je i prije navedeno sve podatke promiješa kako bi oni bili nasumični, a ne uvijek sa istim redoslijedom. Prefetch metoda dohvaća podatke za novu seriju obrade dok se prva još izvodi što

također dovodi do ubrzanja.

```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size = tf.data.AUTOTUNE)
```

Slika 16. Skupovi podataka za treniranje, validaciju i testiranje

Izvor: Izradio autor

Prije treniranja modela slike je dobro provesti kroz razne metode augmentacije kako bi se stvorio bolji skup za treniranje koji će modelu pomoći kasnije kod lakšeg detektiranja bolesti. Metoda augmentacije odnosi se na tehniku uz pomoć koje se može povećati skup podataka. Ona omogućuje stvaranje i preoblikovanje/modificiranje slike koje su već u skupu podataka, što također pomaže kod konvolucijskih neuronskih mreža jer su njima potrebne velike količine podataka kako bi što bolje raspoznavale objekte i slično. (Pai, 2017)

Postoje razne vrste augmentacija, a neke od njih su (Pai, 2017):

- okretanje (*engl. flipping*) - horizontalno i vertikalno kako ne bi dolazilo do pretpostavke da su značajke dostupne samo na jednoj određenoj strani
- skaliranje (*engl. scaling*) – povećavanje ili smanjivanje, korisno je u slučaju kada su slike različitih dimenzija, na primjer s interneta pa ih je sve potrebno skalirati na istu veličinu
- rotacija (*engl. rotation*) – rotacija se koristi kada se želi postići efekt gledanja iz različitih kuteva
- translacija (*engl. translation*) – na ovaj se način objekt pomiče na različite dijelove slike jer se isti može nalaziti na raznim mjestima na slici

Također augmentacijom se smatra i podešavanje svjetline, kontrasta te zasićenosti, a mogu se dodati i razni šumovi kako bi model bio otporniji. (Pai, 2017)

Na Slici 17 mogu se vidjeti dvije metode „RandomFlip“ koja nasumično okreće slike okomito ili naopako ili i jedno i drugo također može se vidjeti metoda „RandomRotation“ koja okreće ili rotira slike za vrijednost od 0.2 ili za 20%.

Augmentacijom podataka modelu je omogućeno da uči na 92460 slika. Do te brojke je došlo množenjem broja slika iz seta za treniranje (3082) i broja epoha (30), To znači da će model vidjeti 30 izmijenjenih verzija svake slike i time si omogućiti efikasnije učenje. U slučaju korištenja 15 epoha model bi imao priliku učiti na 46230 slika.

```
data_augmentation = tf.keras.Sequential([
  RandomFlip("horizontal_and_vertical"),
  RandomRotation(0.2),
])
```

Slika 17. Generiranje novih uzoraka uz pomoć tehnika RandomFlip i RandomRotation

Izvor: Izradio autor

Nakon obrade podataka, sljedeći korak je napraviti model koji je napravljen prema tipičnoj CNN arhitekturi, a može se vidjeti na Slici 18. Prvo se može vidjeti input_shape, a to je ulazni set podataka s dimenzijama slika IMAGE_SIZE x IMAGE_SIZE i brojem kanala koji je postavljen na 3 jer se koriste slike u boji (RGB). Potom slijedi priprema podataka resize_and_rescale uz pomoć koje se ulazne slike postavljaju na željene dimenzije te se skaliraju vrijednosti piksela kako bi se poboljšao proces učenja.

Također, data augmentation, kao što je već navedeno, omogućuje povećanje skupa podataka u smislu da izmjenjuje slike kako bi model poslije bio što precizniji. Primjerice ako se koristi samo RandomFlip, on ima četiri moguća izlaza (horizontalni okret, vertikalni okret, i horizontalni i vertikalni okret ili bez okreta) što znači da će model moći učiti na četiri različite slike umjesto samo na jednoj. Treba napomenuti da su te varijacije nasumične što znači da će nekad biti jedna slika, nekad dvije, a nekad neka druga količina slika.

Nakon što je priprema podataka obavljena slijede konvolucijski slojevi koji postavljaju određeni broj filtera veličine 3x3 na ulaznu sliku i na taj način traže razne teksture, bridove, kutove i male jednostavne uzorke dok ReLU aktivacijska funkcija omogućuje modelu da uči kompliciranije uzorke. Poslije konvolucijskog sloja dolazi sloj sažimanja kojem je svrha smanjiti prostorne dimenzije ulaznih mapa kako bi se smanjilo opterećenje na računalo te se ovaj proces ponavlja nekoliko puta.

Po završetku slojevi se izravnavaju na način da se 3D oblici pretvaraju u 1D vektor koji će služiti kao ulaz u potpuno povezani sloj. Potom potpuno povezani sloj izvodi predviđanje uz pomoć značajki koje su izvučene pomoću konvolucijskih slojeva. Na kraju se se vrši softmax aktivacijska funkcija uz pomoć koje se ispisuje vjerojatnost za svaku klasu.

```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 4

model = models.Sequential([
    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])

model.build(input_shape=input_shape)
```

Slika 18. Izrada modela

Izvor: Izradio autor

```
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
sequential (Sequential)	(32, 256, 256, 3)	0
sequential_1 (Sequential)	(32, 256, 256, 3)	0
conv2d (Conv2D)	(32, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(32, 127, 127, 32)	0
conv2d_1 (Conv2D)	(32, 125, 125, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(32, 62, 62, 64)	0
conv2d_2 (Conv2D)	(32, 60, 60, 64)	36,928
max_pooling2d_2 (MaxPooling2D)	(32, 30, 30, 64)	0
conv2d_3 (Conv2D)	(32, 28, 28, 64)	36,928
max_pooling2d_3 (MaxPooling2D)	(32, 14, 14, 64)	0
conv2d_4 (Conv2D)	(32, 12, 12, 64)	36,928
max_pooling2d_4 (MaxPooling2D)	(32, 6, 6, 64)	0
conv2d_5 (Conv2D)	(32, 4, 4, 64)	36,928
max_pooling2d_5 (MaxPooling2D)	(32, 2, 2, 64)	0
flatten (Flatten)	(32, 256)	0
dense (Dense)	(32, 64)	16,448
dense_1 (Dense)	(32, 4)	260

Total params: 183,812 (718.02 KB)

Trainable params: 183,812 (718.02 KB)

Non-trainable params: 0 (0.00 B)

Slika 19. Detaljna analiza korištene arhitekture

Izvor: Izradio autor

Na Slici 19 može se vidjeti detaljna analiza korištene arhitekture uz pomoć korištenja naredbe model.summary() koja daje informacije o imenima i tipovima svih slojeva, izlaznim oblicima za svaki sloj te daje informacije o broju težinskih parametara svakog sloja. Na slici se može vidjeti da je nakon prve konvolucije veličina slike smanjena sa (32, 256, 256, 3) na (32, 254, 254, 32), a također se može vidjeti da je nakon prvog sloja sažimanja slika smanjena na (32, 127, 127, 32) te se na taj način smanjuje uzrokovanje slike.

Nakon definiranja arhitekture mreže, sljedeći koraci su postavljanje Adam optimizacije i funkcije gubitka kao što se može vidjeti na Slici 20.

```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

Slika 20. Prikaz postavljanja Adam optimizacije u kodu

Izvor: Izradio autor

Adam (*engl. Adaptive Moment Estimation*) je široko korišten optimizacijski algoritam u dubokom učenju, a posebno za učenje konvolucijskih neuronskih mreža. Adam koristi dva druga također popularna optimizatora AdaGrad i RMSProp kako bi osigurao što veću efikasnost i brzinu učenja modela. Još jedna strana njegove učinkovitosti je ta što prilagođava stopu učenja za svaki parametar zasebno, umjesto da koristi jednu stalnu stopu učenja. Njegova zadana stopa učenja je 0.001. Jako je efikasan kada je potrebno raditi s velikim količinama podataka ili parametara. Također zahtijeva manje memorije i jako je učinkovit. (What is Adam Optimizer?, 2024)

Na Slici 20 također se može vidjeti funkcija gubitka koja je temeljna komponenta u strojnom i dubokom učenju, a koristi se za procjenu koliko su dobro predviđanja modela usklađena sa stvarnim ciljanim vrijednostima. Ona prikazuje razliku između predviđenog izlaza modela i stvarnog izlaza te na taj način usmjerava proces optimizacije i postupno smanjuje tu razliku na minimum. To znači da tijekom treniranja model daje neka predviđanja nad ulaznim podatcima i funkcija gubitka izračunava gubitak uspoređujući krajnje ili stvarne rezultate s predviđenima. Potom u ovom slučaju Adam optimizator koristi te rezultate kako bi smanjio greške i gubitke u nadolazećim krugovima. (What is Adam Optimizer?, 2024)

U ovom se radu koristi Sparse Categorical Crossentropy funkcija gubitka u svrhu rješavanja zadataka klasifikacije više klasa. Ova funkcija izračunava negativnu logaritamsku vjerojatnost prave klase gledajući na predviđenu distribuciju vjerojatnosti po klasama, a u cilju joj je minimizirati gubitak na način da predviđenu vrijednost napravi što bližom jedinici kako bi model bio što točniji. Vrlo je korisna u trenutcima kada postoji veliki broj klasa (Rahmna, 2023).

Na Slici 21 može se vidjeti proces pokretanja treniranja modela.

```
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=15,
)
```

Slika 21. Pokretanje procesa treniranja modela

Izvor: Izradio autor

7. Rezultati i diskusija

Provedenim testiranjem može se vidjeti da je točnost rasla s prve epohe koja je imala 48,48% do zadnje epohe kojoj je točnost došla do 94,50% (Slika 22). Također uz pomoć evaluate metode, na modelu se mogu provesti i slike iz testnog skupa podataka koje model nije imao prilike vidjeti te se može vidjeti da mu je točnost na novim podatcima 97,96% (Slika 23).

```
Epoch 1/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 74s 626ms/step - accuracy: 0.4848 - loss: 1.0693 - val_accuracy: 0.8099 -
val_loss: 0.4801
Epoch 2/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 53s 553ms/step - accuracy: 0.8661 - loss: 0.3350 - val_accuracy: 0.8672 -
val_loss: 0.2753
Epoch 3/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 53s 549ms/step - accuracy: 0.8371 - loss: 0.4266 - val_accuracy: 0.8828 -
val_loss: 0.2242
Epoch 4/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 52s 543ms/step - accuracy: 0.9085 - loss: 0.1969 - val_accuracy: 0.9323 -
val_loss: 0.1729
Epoch 5/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 523ms/step - accuracy: 0.9234 - loss: 0.1852 - val_accuracy: 0.9036 -
val_loss: 0.2140
Epoch 6/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 519ms/step - accuracy: 0.9233 - loss: 0.1907 - val_accuracy: 0.9479 -
val_loss: 0.1441
Epoch 7/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 520ms/step - accuracy: 0.9342 - loss: 0.1546 - val_accuracy: 0.9401 -
val_loss: 0.1229
Epoch 8/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 52s 542ms/step - accuracy: 0.9365 - loss: 0.1405 - val_accuracy: 0.9375 -
val_loss: 0.1254
Epoch 9/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 518ms/step - accuracy: 0.9488 - loss: 0.1346 - val_accuracy: 0.9193 -
val_loss: 0.2686
Epoch 10/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 51s 532ms/step - accuracy: 0.9239 - loss: 0.1810 - val_accuracy: 0.9219 -
val_loss: 0.1872
Epoch 11/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 54s 568ms/step - accuracy: 0.9328 - loss: 0.1575 - val_accuracy: 0.9453 -
val_loss: 0.1144
Epoch 12/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 55s 576ms/step - accuracy: 0.9471 - loss: 0.1447 - val_accuracy: 0.9557 -
val_loss: 0.0971
Epoch 13/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 517ms/step - accuracy: 0.9506 - loss: 0.1166 - val_accuracy: 0.9557 -
val_loss: 0.0844
Epoch 14/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 49s 515ms/step - accuracy: 0.9650 - loss: 0.0974 - val_accuracy: 0.9193 -
val_loss: 0.1971
Epoch 15/15
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 516ms/step - accuracy: 0.9450 - loss: 0.1257 - val_accuracy: 0.9688 -
val_loss: 0.0696
```

Slika 22. Proces učenja s 15 epoha

Izvor: Izradio autor

```
scores = model.evaluate(test_ds)
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 151ms/step - accuracy: 0.9796 - loss: 0.0561
```

Slika 23. Testiranje modela na podatcima iz skupa podataka za testiranje

Izvor: Izradio autor

Nakon provedenog testiranja modela treniranog s 15 epoha, provedeno je još jedno testiranje gdje je model treniran s 30 epoha (Slika 24) s ciljem pronalaska odgovora na pitanje hoće li više epoha dovesti do veće točnosti. Unatoč dvostruko većem broju epoha proces učenja je uspio doći na 94,92% točnosti s početnih 45,35% te je na testnom skupu podataka pokazao 90,32% točnosti.

```
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 520ms/step - accuracy: 0.9351 - loss: 0.1506 - val_accuracy: 0.9427 - val_loss: 0.1233
Epoch 16/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 516ms/step - accuracy: 0.9379 - loss: 0.1714 - val_accuracy: 0.9557 - val_loss: 0.1126
Epoch 17/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 51s 534ms/step - accuracy: 0.9430 - loss: 0.1583 - val_accuracy: 0.9427 - val_loss: 0.1275
Epoch 18/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 52s 537ms/step - accuracy: 0.9362 - loss: 0.1668 - val_accuracy: 0.9479 - val_loss: 0.1198
Epoch 19/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 519ms/step - accuracy: 0.9458 - loss: 0.1493 - val_accuracy: 0.9167 - val_loss: 0.1904
Epoch 20/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 517ms/step - accuracy: 0.9316 - loss: 0.1713 - val_accuracy: 0.9297 - val_loss: 0.1585
Epoch 21/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 518ms/step - accuracy: 0.9351 - loss: 0.1618 - val_accuracy: 0.9167 - val_loss: 0.2137
Epoch 22/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 519ms/step - accuracy: 0.9316 - loss: 0.1840 - val_accuracy: 0.9688 - val_loss: 0.0979
Epoch 23/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 518ms/step - accuracy: 0.9518 - loss: 0.1310 - val_accuracy: 0.9505 - val_loss: 0.1037
Epoch 24/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 519ms/step - accuracy: 0.9473 - loss: 0.1447 - val_accuracy: 0.9557 - val_loss: 0.1115
Epoch 25/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 518ms/step - accuracy: 0.9401 - loss: 0.1392 - val_accuracy: 0.9427 - val_loss: 0.1165
Epoch 26/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 518ms/step - accuracy: 0.9302 - loss: 0.1797 - val_accuracy: 0.9401 - val_loss: 0.1501
Epoch 27/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 519ms/step - accuracy: 0.9453 - loss: 0.1405 - val_accuracy: 0.9557 - val_loss: 0.0976
Epoch 28/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 519ms/step - accuracy: 0.9466 - loss: 0.1391 - val_accuracy: 0.9714 - val_loss: 0.0785
Epoch 29/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 517ms/step - accuracy: 0.9456 - loss: 0.1291 - val_accuracy: 0.9740 - val_loss: 0.0857
Epoch 30/30
96/96 ━━━━━━━━━━━━━━━━━━━━ 50s 523ms/step - accuracy: 0.9492 - loss: 0.1199 - val_accuracy: 0.9219 - val_loss: 0.1818

]: scores = model.evaluate(test_ds)
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 132ms/step - accuracy: 0.9032 - loss: 0.1844
```

Slika 24. Proces učenja s 30 epoha

Izvor: Izradio autor

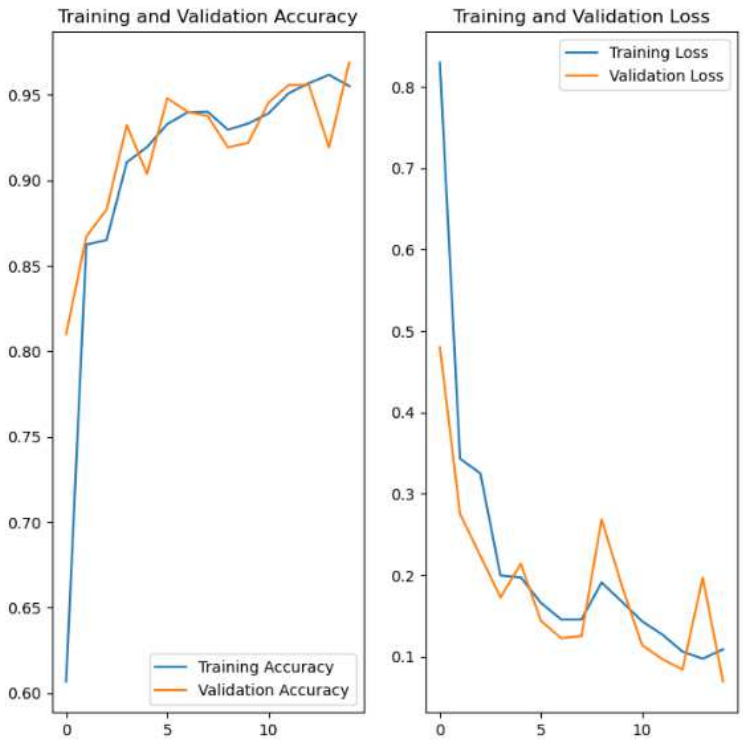
Na Slici 25 može se vidjeti kod uz pomoću kojeg su napravljeni grafovi koji prikazuju točnost i vrijednost funkcije gubitka tijekom treniranja modela. Kod stvara polje od 8x8 inča te ga rastavlja na jedan red i dva stupca. U prvom stupcu (lijevo) se može vidjeti graf koji prikazuje točnost modela. Na sličan način funkcionira i graf na desnoj strani samo ovaj prikazuje vrijednosti funkcije gubitka.

```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(range(EPOCHS), acc, label='Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(range(EPOCHS), loss, label='Training Loss')
plt.plot(range(EPOCHS), val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Slika 25. Kod koji iscrtava grafove za točnost i vrijednost funkcije gubitka tijekom treniranja

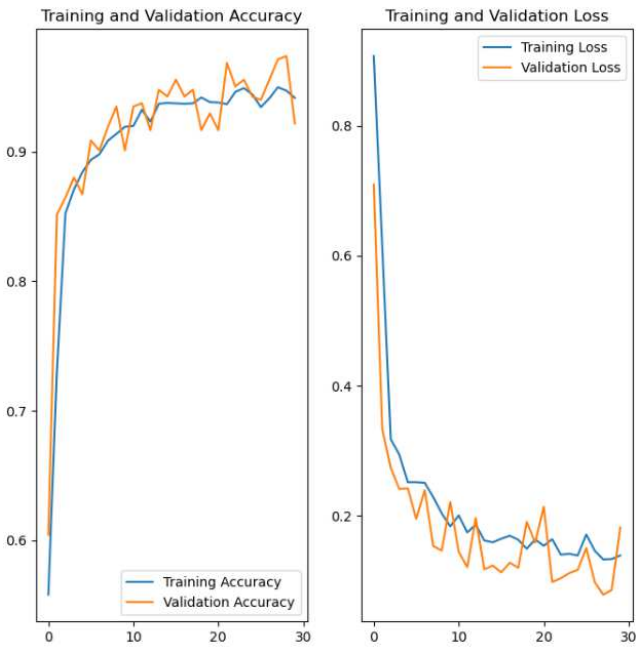
Izvor: Izradio autor

Na Slici 26 na lijevom grafu može se vidjeti da je model došao do visoke točnosti već na četvrtoj ili petoj epohi što se također može vidjeti i na Slici 27 gdje je korišteno 30 epoha. S desne strane na obje slike može se vidjeti da je vrijednost funkcije gubitka sve manja, a najveći pad doživi već na drugoj ili trećoj epohi.



Slika 26. Graf prikazuje točnost i vrijednost funkcije gubitka tijekom treniranja modela s 15 epoha

Izvor: Izradio autor



Slika 27. Graf prikazuje točnost i vrijednost funkcije gubitka tijekom treniranja modela s 30 epoha

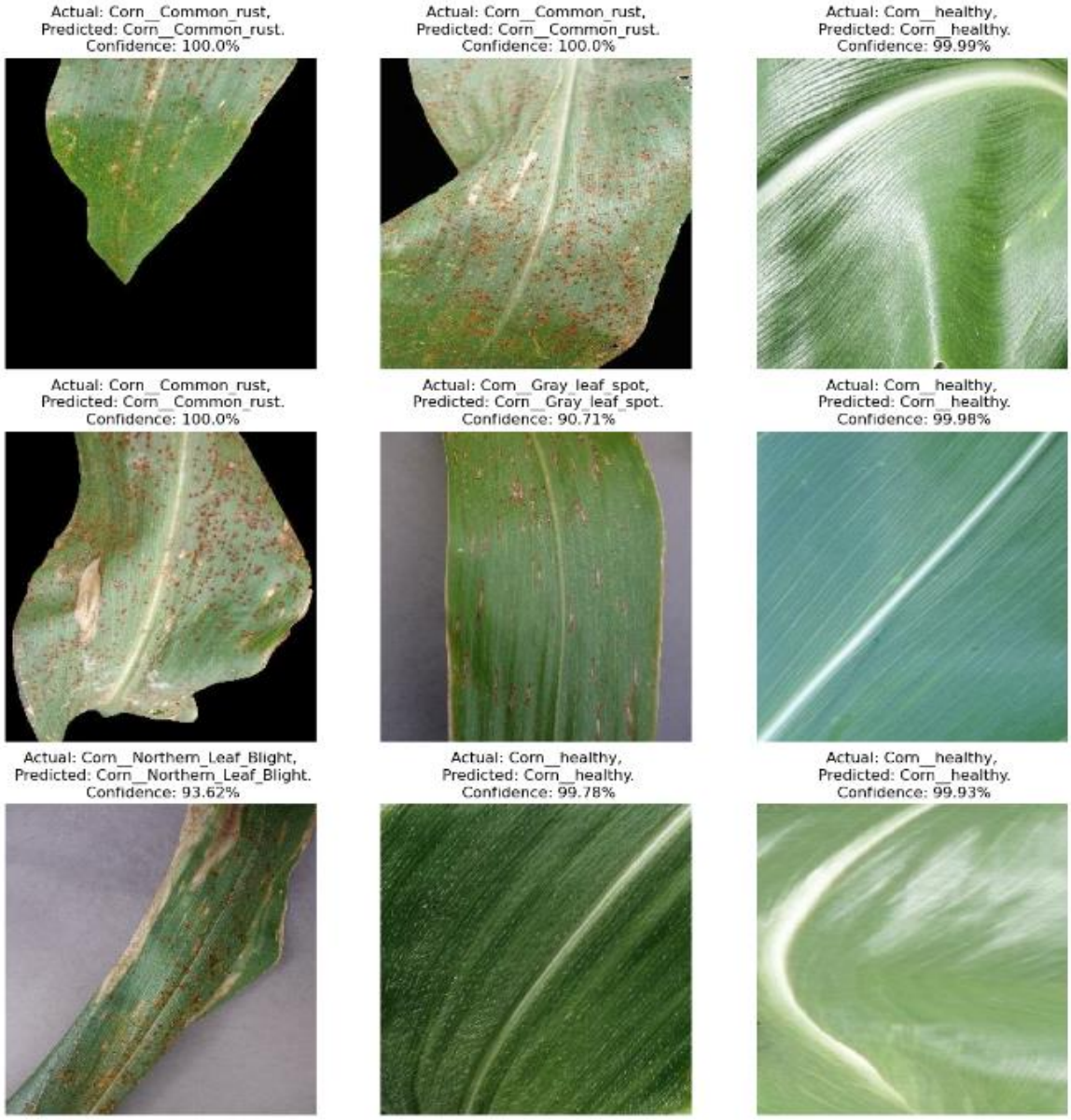
Izvor: Izradio autor

Unatoč uglavnom dobrim predikcijama ponekad može doći i do lošije pouzdanosti te time i do krive predikcije. Na Slici 28 prikazan je primjer jedne takve pogrešne predikcije gdje je model pogrešno predvidio da se radi o bolesti sive pjegavosti, umjesto bolesti sjeverne plamenjače o kojoj se zapravo radi. Do krive pretpostavke je moglo doći zbog sličnosti između navedenih bolesti jer obje imaju smećkasto žute ili bež boje na zaraženom listu kao što je vidljivo sa Slike 28.

Slika 28. Primjer pogrešne predikcije modela

Izvor: Izradio autor

Na kraju, provedena je još jedna analiza na skupu podataka za treniranje u kojoj se može vidjeti devet slika s njihovim imenima, predikcijama i pouzdanosti (*engl. confidence*) koja će u postotku reći koliko je model siguran da je točno predvidio rezultat. Na temelju rezultata koji variraju od 90% do 100% posto pouzdanosti može se zaključiti da je model uspješan u otkrivanju i raspoznavanju bolesti koje su obrađene u ovom radu. Također to znači da bi ovakav model bio uspješan u aplikacijama koje bi poljoprivrednicima omogućile rano otkrivanje bolesti lista kukuruza i samim time velike uštede. Primjer rezultata provedene analize pouzdanosti može se vidjeti na Slici 29



Slika 29. Primjeri ispravnih predikcija modela uz vrijednosti pouzdanosti predikcije

Izvor: Izradio autor

Na Slici 29 mogu se vidjeti sve bolesti iz skupa podataka (sjeverna plamenjača, hrđa, siva pjegavost) te zdravi list kukuruza. Prvo što se može uočiti su pravi rezultati slika listova kukuruza koje su ispod naljepnice pod nazivom „Actual“, zatim ispod toga može se vidjeti naljepnica „Predicted“ koja govori što je model prvotno predvidio i mislio. Na kraju se može vidjeti naljepnica „Confidence“ koja u postotku govori koliko je model pouzdan u to da je napravio točnu predikciju.

8. Zaključak

Ovaj diplomski rad istražuje primjenu konvolucijskih neuronskih mreža za klasifikaciju bolesti lista kukuruza na digitalnim slikama. Korištenjem CNN-a, uspješno je razvijen model koji može prepoznati i klasificirati listove zdravog kukuruza te tri vrste bolesti, a one su: siva pjegavost, hrđa lista kukuruza i sjeverna plamenjača. Provedenim istraživanjem i testiranjem pokazalo se da je model postigao visoku točnost klasifikacije (97,96%) nad podacima koji nisu korišteni u fazi treniranja.

Tijekom rada, korišteni su različiti alati i biblioteke poput Pythona, TensorFlow-a, Numpy-a i Matplotliba, koji su omogućili efikasnu obradu podataka i razvoj modela. Skup podataka „PlantVillage Dataset“ poslužio je kao kvalitetan temelj za treniranje i testiranje modela.

Iz ovih rezultata može se zaključiti da konvolucijske neuronske mreže mogu biti vrlo učinkovite u rješavanju problema prepoznavanja biljnih bolesti, u ovom slučaju bolesti lista kukuruza i na taj način pružaju poljoprivrednicima alat za rano otkrivanje i prevenciju bolesti, što može rezultirati značajnim ekonomskim uštedama i povećanjem kvalitete kukuruza i prinosa usjeva.

U budućnosti bi se mogla provoditi daljnja istraživanja koja bi se fokusirala na proširenje modela koji bi mogli prepoznavati i druge bolesti lista kukuruza, a također isti princip bi se mogao primijeniti i nad drugim biljkama. Također, upotreba naprednijih tehnika obrade slike i optimizacije modela mogla bi dodatno povećati učinkovitost i pouzdanost sustava. U zaključku, ovaj rad pokazuje da primjena konvolucijskih neuronskih mreža u poljoprivredi ima velik potencijal i može značajno doprinijeti unapređenju tehnika upravljanja usjevima i održivom razvoju poljoprivrede.

Popis literature

1. *8 Common Types of Neural Networks.* 23. Travanj 2024. https://www.coursera.org/in/articles/types-of-neural-networks.
2. *A Concise History of Neural Networks.* 14. kolovoz 2016. https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec.
3. Campesato, Oswald. *Artificial Intelligence, Machine Learning and Deep Learning.* MERCURY LEARNING AND INFORMATION, 2020.
4. *CPSC 440: Machine Learning.* 2022. https://www.cs.ubc.ca/~schmidtm/Courses/440-W22/L8.pdf.
5. Géron, Aurélien. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.* O'Reilly Media, Inc., 2019.
6. How, Khoh Wee. *Breast cancer classification with histopathological image based on machine learning.* listopad 2023. https://www.researchgate.net/figure/Max-and-average-pooling-21_fig2_374352652.
7. *Introduction to TensorFlow.* n.d. https://www.tensorflow.org/learn.
8. *Maize Diseases: A Guide for Field Identification.* The CIMMYT Maize Program, 2004.
9. *Matplotlib Tutorial.* 2024. https://www.w3schools.com/python/matplotlib_intro.asp.
10. Naebi, Ahmad, i Zuren Feng. *The Performance of a Lip-Sync Imagery Model, New Combinations of Signals, a Supplemental Bond Graph Classifier, and Deep Formula Detection as an Extraction and Root Classifier for Electroencephalograms and Brain–Computer Interfaces.* 2023. https://www.mdpi.com/2076-3417/13/21/11787.
11. *NumPy documentation.* 2024. https://numpy.org/doc/stable/.
12. Özer, Anil. *A Brief History of the Neural Networks.* 20. listopad 2023. https://www.kdnuggets.com/a-brief-history-of-the-neural-networks.
13. Pai, Prasad. *Data Augmentation Techniques in CNN using Tensorflow.* 10. listopad 2017. https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9.
14. *PlantVillage Dataset.* n.d. https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset.
15. *Python Introduction.* 2024. https://www.w3schools.com/python/python_intro.asp.
16. Rahmna, Moklesur. *What You Need to Know about Sparse Categorical Cross Entropy.* 3. ožujak 2023. https://rmoklesur.medium.com/what-you-need-to-know-about-sparse-categorical-cross-entropy-9f07497e3a6f.
17. Sewak, Mohit, Rezaul Karim, i Pradeep Pujari. *Practical Convolutional Neural Networks.* Packt Publishing, 2018.
18. *Softmax vs LogSoftmax.* 10. listopad 2021. https://medium.com/@AbhiramiVS/softmax-vs-logsoftmax-eb94254445a2.
19. *Tanh Activation Function.* n.d. https://www.ml-science.com/tanh-activation-function.
20. *The Python Tutorial.* 30. Kolovoz 2024. https://docs.python.org/3/tutorial/index.html.
21. Toprak, Mehmet. *Activation Functions for Deep Learning.* 14. lipanj 2020. https://medium.com/@toprak.mhmt/activation-functions-for-deep-learning-13d8b9b20e.

22. Trask, Andrew W. *grokking Deep Learning.* Manning Publications Co., 2019.
23. *Types of Neural Networks and Definition of Neural Network.* 25. Srpanj 2024. https://www.mygreatlearning.com/blog/types-of-neural-networks/#h-modular-neural-network.
24. *Using Matplotlib.* 2024. https://matplotlib.org/stable/users/index.
25. Vaishya, Ayush. *Mastering OpenCV with Python, Use NumPy, Scikit, TensorFlow, and Matplotlib to learn Advanced algorithms for Machine Learning through a set of Practical Projects.* Delhi: Orange Education Pvt Ltd, 2023.
26. *What is a neural network?* n.d. https://www.cloudflare.com/learning/ai/what-is-neural-network/.
27. *What is a Neural Network?* n.d. https://aws.amazon.com/what-is/neural-network/.
28. *What is Adam Optimizer?* 20. ožujak 2024. https://www.geeksforgeeks.org/adam-optimizer/.

Popis slika

Slika 1. Odnosi između umjetne inteligencije, strojnog učenja i dubokog učenja 4
Slika 2. Osnovni prikaz slojeva duboke neuronske mreže 6
Slika 3. Tipična konvolucijska neuronska mreža 8
Slika 4. Prikaz funkcija sloja sažimanja, Max Pooling i Average Pooling 10
Slika 5. Graf sigmoidne aktivacijske funkcije 11
Slika 6. Graf aktivacijske funkcije hiperboličke tangente 12
Slika 7. Graf ReLu aktivacijske funkcije 12
Slika 8. Graf softmax aktivacijske funkcije 13
Slika 9. Siva pjegavost lista kukuruza 15
Slika 10. Hrđa lista kukuruza 15
Slika 11. Sjeverna plamenjača na listu kukuruza 16
Slika 12. Popis Python biblioteka korištenih u radu 16
Slika 13. Postavljanje parametara 20
Slika 14. Unos podataka 20
Slika 15. Prikaz slika zdravih i bolesnih listova sa njihovim pripadajućim imenima .. 21
Slika 16. Skupovi podataka za treniranje, validaciju i testiranje 22
Slika 17. Generiranje novih uzoraka uz pomoć tehnika RandomFlip i RandomRotation 23
Slika 18. Izrada modela 24
Slika 19. Detaljna analiza korištene arhitekture 25
Slika 20. Prikaz postavljanja Adam optimizacije u kodu 26
Slika 21. Pokretanje procesa treniranja modela 27
Slika 22. Proces učenja s 15 epoha 28
Slika 23. Testiranje modela na podatcima iz skupa podataka za testiranje 29
Slika 24. Proces učenja s 30 epoha 29
Slika 25. Kod koji iscrtava grafove za točnost i vrijednost funkcije gubitka tijekom treniranja 30
Slika 26. Graf prikazuje točnost i vrijednost funkcije gubitka tijekom treniranja modela s 15 epoha 30
Slika 27. Graf prikazuje točnost i vrijednost funkcije gubitka tijekom treniranja modela s 30 epoha 31
Slika 28. Primjer pogrešne predikcije modela 31

Slika 29. Primjeri ispravnih predikcija modela uz vrijednosti pouzdanosti predikcije 32

Popis tablica

Tablica 1. Prikaz koraka u konvolucijskom sloju 9

Sažetak

Ovaj diplomski rad bavi se primjenom konvolucijskih neuronskih mreža (CNN) za klasifikaciju bolesti lista kukuruza putem obrade digitalnih slika. Rad je usmjeren na razvoj modela dubokog učenja koristeći CNN za prepoznavanje i klasifikaciju triju specifičnih bolesti lista kukuruza: sive pjegavosti, hrđe i sjeverne plamenjače, uz pomoć javno dostupnog skupa slikovnih podataka. CNN je odabran zbog svoje sposobnosti prepoznavanja uzoraka u slikama, što ga čini idealnim za klasifikaciju biljnih bolesti. Na početku je prikazan teorijski okvir strojnog i dubokog učenja, uključujući detaljniju obradu arhitekture CNN-a koja se sastoji od konvolucijskih slojeva, slojeva sažimanja te potpuno povezanih slojeva, uz osvrt na specifične aktivacijske funkcije. Proces izrade modela počinje prikupljanjem podataka, gdje su slike kukuruza s bolestima pripremljene i podijeljene na skupove za treniranje, validaciju i testiranje. Primijenjena je augmentacija podataka radi povećanja broja uzoraka, čime je model treniran na većem broju slika u različitim varijantama. CNN arhitektura implementirana u Python programskom okruženju korištenjem Tensorflowa pokazala je visoku točnost tijekom testiranja, dosegnuvši 97,96% točnosti na testnim podatcima. Dobiveni rezultati pokazali su visoku pouzdanost modela u prepoznavanju bolesti, uz manja odstupanja u specifičnim slučajevima zbog velike sličnosti između bolesti. Zaključno, CNN se pokazao vrlo učinkovitim u klasifikaciji bolesti lista kukuruza te ima potencijal za daljnju primjenu u poljoprivredi za druge vrste usjeva.

Ključne riječi: konvolucijske neuronske mreže, bolesti kukuruza, umjetna inteligencija, klasifikacija slika, duboko učenje

Abstract

This thesis deals with the application of convolutional neural networks (CNNs) for the classification of corn leaf diseases through digital image processing. The work is focused on the development of a deep learning model using CNN for the recognition and classification of three specific corn leaf diseases: gray leaf spot, rust, northern leaf blight, with the help of a publicly available image dataset. CNN was chosen for its ability to recognize patterns in images, which makes it ideal for plant disease classification. At the beginning, the theoretical framework of machine learning and deep learning is presented, including a more detailed treatment of the CNN architecture consisting of convolutional layers, pooling layers and fully connected layers, with reference to specific activation functions. The model development process begins with data collection, where images of corn with diseases are prepared and divided into training, validation, and testing sets. Data augmentation was applied to increase the number of samples, allowing the model to be trained on a larger number of images in various variants. The CNN architecture, implemented in the Python programming environment using Tensorflow, showed high accuracy during testing, reaching 97.96% accuracy on the test data. The obtained results demonstrated the model's high reliability in recognizing diseases, with minor deviations in specific cases due to the significant similarity between diseases. In conclusion, CNN proved to be very effective in the classification of corn leaf diseases and has the potential for further application in agriculture for other types of crops.

Keywords: convolutional neural networks, corn diseases, artificial intelligence, image classification, deep learning