SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1393

MOBILNA APLIKACIJA ZA KLASIFIKACIJU VRSTE CVIJEĆA SA SLIKE KORIŠTENJEM METODA DUBOKOG UČENJA

Luka Raić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1393

MOBILNA APLIKACIJA ZA KLASIFIKACIJU VRSTE CVIJEĆA SA SLIKE KORIŠTENJEM METODA DUBOKOG UČENJA

Luka Raić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1393

Pristupnik:	**Luka Raić (0036537547)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Alan Jović

Zadatak: **Mobilna aplikacija za klasifikaciju vrste cvijeća sa slike korištenjem metoda dubokog učenja**

Opis zadatka:

Ponekad nije jednostavno odrediti vrstu cvijeća, pogotovo ako se radi o nekoj vrsti koja nije uobičajena. Stoga bi usluga automatizirane klasifikacije vrste cvijeća bila vrlo korisna većem broju potencijalnih korisnika. Cilj ovog završnog rada je izrada mobilne aplikacije koja će ponuditi visokotočnu klasifikaciju vrste cvijeća sa slike korištenjem metoda dubokog učenja. Pritom je za učenje klasifikacijskog modela potrebno koristiti slobodno dostupni skup podataka sa slikama cvijeća, kao što je Oxford 102 Flower (https://paperswithcode.com/dataset/oxford-102-flower). Potrebno je pretpostaviti da se na slici nalazi samo jedna vrsta cvijeća. Za arhitekturu klasifikacijskog modela potrebno je isprobati uobičajene arhitekture dubokog učenja za rad sa slikama, kao što su konvolucijske neuronske mreže i njihove varijacije. Mobilna aplikacija treba imati ugrađen najtočniji izgrađeni model dubokog učenja za klasifikaciju vrste cvijeća i treba ponuditi jednostavnu uslugu u kojoj će se na temelju uslikane slike cvijeta odrediti o kojoj vrsti cvijeta se radi i ta će se informacija prikazati korisniku. Implementaciju mobilne aplikacije i izgradnju modela dubokog učenja potrebno je napraviti u programskim jezicima po vlastitom izboru, a za izgradnju modela može se u slučaju nedostatka vlastitih sklopovskih resursa koristiti dostupna web rješenja (npr. Google Colab).

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem se svom mentoru, izv. prof. dr. sc. Alanu Joviću, na pomoći pri izradi završnog rada te svojoj majci Jelici, ocu Robertu i sestri Karli na podršci.

Sadržaj

Uvod

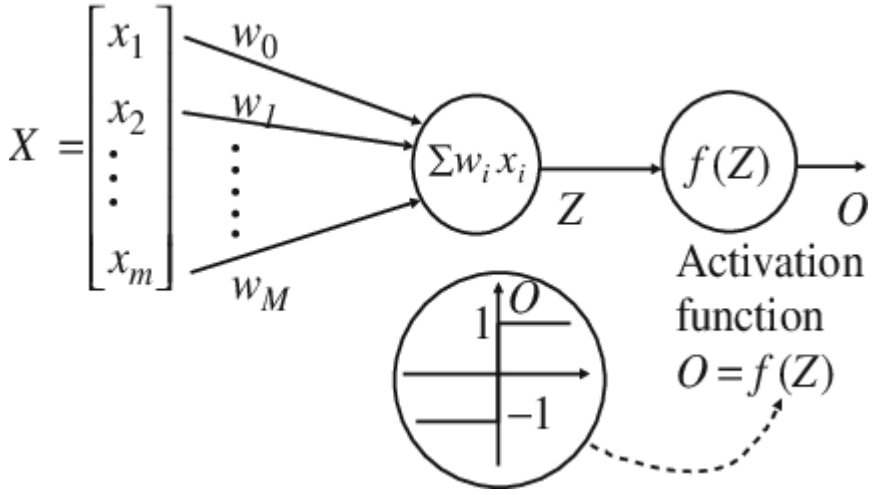
U svijetu s bogatom florom, lako je izgubiti se u brojnim vrstama cvijeća. Poznavanje njihovih imena i karakteristika može biti izuzetno izazovno, a pogotovo za one koji se ne bave botanikom kao strukom. U ovom radu predstavljen je Florify, aplikacija koja će svim ljubiteljima cvijeća omogućiti da s lakoćom identificiraju vrstu cvijeta čija slika im je pri ruci.

Florify je mobilna aplikacija namijenjena za operacijski sustav Android koja pruža uslugu automatizirane klasifikacije vrste cvijeća pomoću ugrađenog modela dubokog učenja. Nakon pokretanja aplikacije korisnik može ili uslikati cvijet ili odabrati postojeću sliku cvijeta iz galerije uređaja te se ta slika prosljeđuje klasifikacijskom modelu temeljenom na konvolucijskoj neuronskoj mreži. Model predanu sliku prikladno predobrađuje, klasificira i vraća rezultat na temelju kojeg aplikacija ispisuje tri najvjerojatnija razreda kojima bi uslikani cvijet mogao pripadati te pripadne sigurnosti modela u svoje pogotke.

U nastavku rada će se za početak govoriti o samim umjetnim neuronskim mrežama te će se proći kroz teoriju koja stoji iza dvije često korištene arhitekture: višeslojnog perceptrona i konvolucijske neuronske mreže. Nakon toga predstavit će se podatkovni skup korišten u procesu učenja razvijenih modela, arhitekture samih modela te primijenjeni postupci predobrade ulaznih slika. Priča o modelima dubokog učenja završit će se predstavljanjem rezultata njihovog učenja te će se za kraj predstaviti sama mobilna aplikacija, njezina struktura, arhitektura te ponašanje i izgled korisničkog sučelja za određeni ulaz.

1. Umjetne neuronske mreže

Kada su u pitanju izumi, vidljivo je da je čovjek oduvijek svoju inspiraciju pronalazio u prirodi: kako kod solarnih panela koje, kao i biljke kroz proces fotosinteze, pretvaraju energiju svjetlosti u kemijsku energiju, ili pak vlakova velikih brzina u japanskom Shinkansenu čiji je dizajn inspiriran kljunom vodomara, tako i kod zrakoplova čiji su dizajn braća Wright napravili početkom 20. stoljeća nakon što su promatrali let goluba. S obzirom na to da je kod ljudi oduvijek postojala želja da naprave kopiju sebe i stvore umjetan život (*Frankenstein; or, The Modern Prometheus*, *I, Robot*), bilo je samo pitanje vremena kada će na red doći ljudski mozak. Do toga je i došlo 1943. godine, kada su američki znanstvenici Warren McCulloch i Walter Pitts u svom radu *A Logical Calculus of Ideas Immanent in Nervous Activity* predstavili prvi model biološkog neurona prikazanog na slici 1.1.

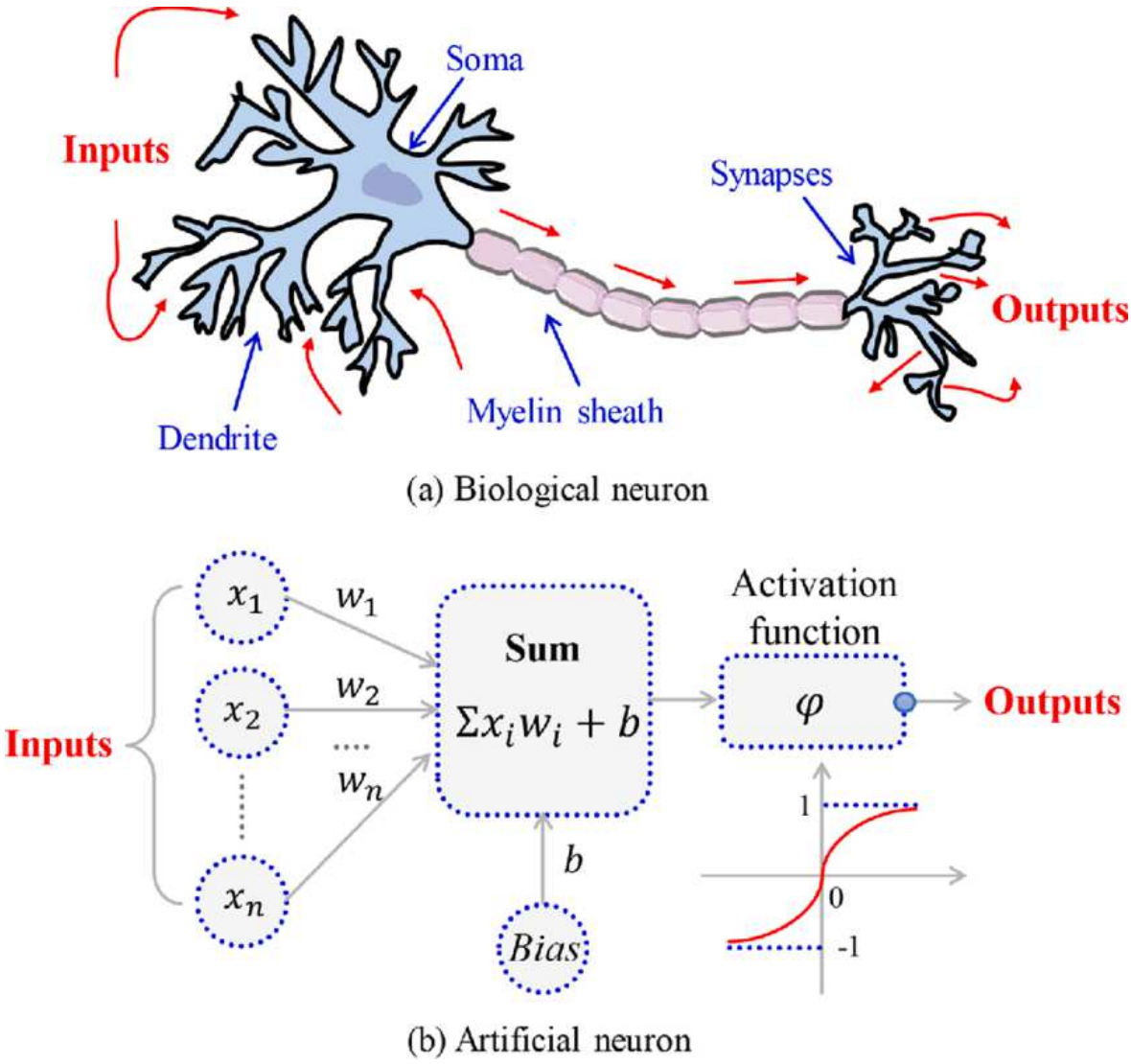


Slika 1.1 Model neurona McCulloch-Pitts [1]

Umjetna neuronska mreža (engl. *artificial neural network*, ANN) je sustav međusobno povezanih elemenata nazvani umjetni neuroni koji služi za raznovrsna izračunavanja, zasnovan na pokušaju oponašanja rada ljudskoga mozga [2]. Svaki umjetni neuron prima signale od nekih drugih neurona, adekvatno ih obrađuje i zatim šalje signal opet nekim drugim neuronima. Ulazni signali su zapravo neke realne vrijednosti, a izlazni signal se dobije tako da se sumi ulaza pridoda vrijednost koja se naziva pragom (engl. *bias*) i taj zbroj se propusti kroz tzv. prijenosnu funkciju. Pritom je važno imati na umu da je suma ulaznih vrijednosti težinska – svakoj vezi između dva neurona je pridružen realni broj koji se naziva težinskim faktorom (engl. *weight*), pri čemu se te vrijednosti korigiraju tijekom

procesa učenja umjetne neuronske mreže kako bi se poboljšale njezine performanse na zadatku koji obavlja.

Vidljivo je da je prethodno opisani način funkcioniranja umjetnog neurona analogan biološkom: kod bioloških neurona, dendriti primaju signale koje ili pojačavaju ili prigušuju, signali se u somi neurona sumiraju te se rezultantni signal, ako prelazi određenu vrijednost, šalje putem aksona i sinapsi do drugih neurona [3]. Sličnosti umjetnog i biološkog neurona vizualno su prikazane na slici 1.2.



Slika 1.2 Usporedba biološkog i umjetnog neurona [4]

Umjetne neuronske mreže mogu učiti iz iskustva te izvući zaključke iz kompleksnog i naizgled nepovezanog skupa informacija te se zbog toga koriste u rješavanju različitih problema, a jedan od njih je upravo klasifikacija slika. U nastavku će fokus biti prebačen na

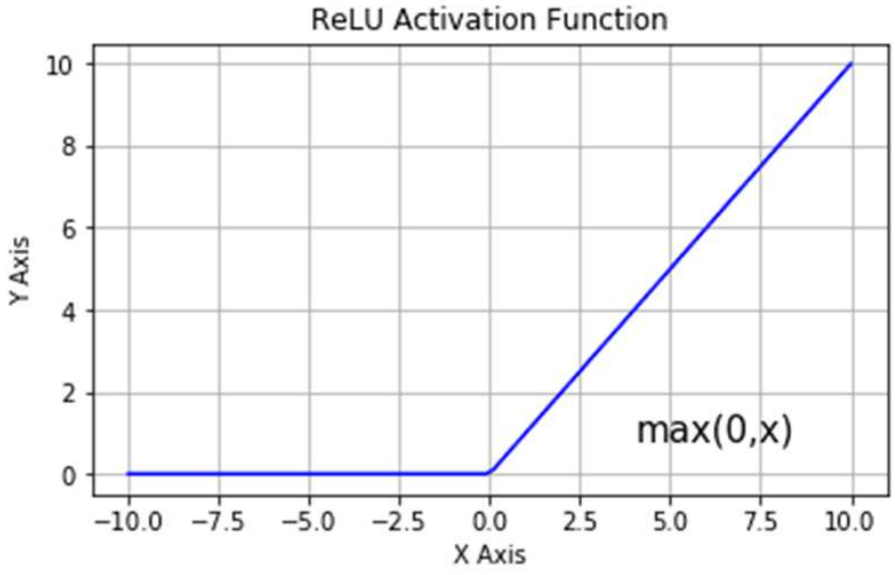
dvije arhitekture umjetnih neuronskih mreža koje omogućuju rješavanje ovog problema, a to su višeslojni perceptron i konvolucijska neuronska mreža.

1.1. Višeslojni perceptron

Višeslojni perceptron (engl. *multilayer perceptron*, MLP) je umjetna neuronska mreža koja se sastoji od neurona organiziranih u minimalno tri sloja, pri čemu se razlikuju:

- ulazni sloj
- skriveni sloj (može ih biti više)
- izlazni sloj

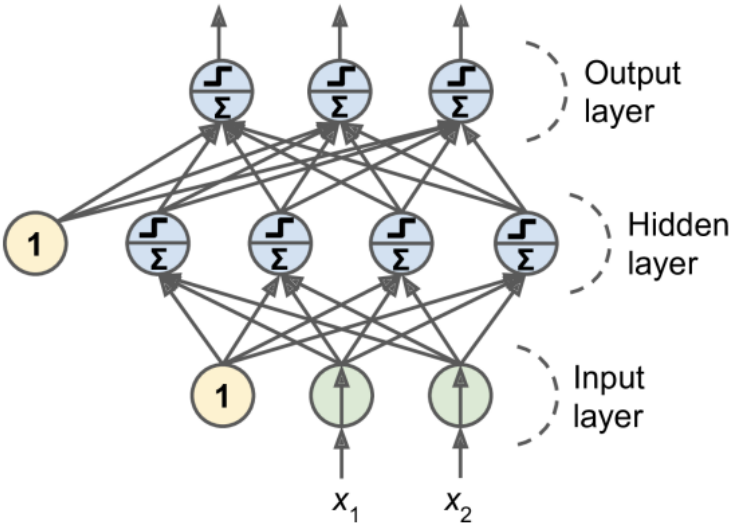
U ulaznom sloju se nalaze neuroni koji na svom ulazu primaju ulazni podatak te ga prosljeđuju na svoj izlaz bez ikakve obrade: npr. u kontekstu rješavanja problema klasifikacije crno-bijelih slika, svaki ulazni neuron sadržava vrijednost koja predstavlja svjetlinu pojedinog piksela ulazne slike. Neuroni skrivenih slojeva na ulaz primaju izlazne vrijednosti ulaznog sloja ili nekog drugog skrivenog sloja te zbroj težinske sume ulaznih vrijednosti i praga provlače kroz prijenosnu funkciju. S obzirom na to da je poželjno povećati ekspresivnost mreže, često se biraju prijenosne funkcije koje su nelinearne: najčešće je u pitanju neka sigmoidna funkcija poput tangensa hiperbolnog ili logističke funkcije, a u zadnje vrijeme se često koristi i ReLU (engl. *rectified linear unit*), čiji graf je vidljiv na slici 1.3.



Slika 1.3 Prijenosna funkcija ReLU [5]

U izlaznom sloju neuroni na ulaz primaju izlazne vrijednosti neurona prethodnog skrivenog sloja te ih obrađuju na jednak način kao i neuroni skrivenog sloja, s tim da izbor prijenosne funkcije ovisi o problemu koji perceptron rješava (npr. kod rješavanja problema binarne klasifikacije čest izbor je logistička funkcija, kod rješavanja problema klasifikacije s više od dva moguća razreda koristi se funkcija *softmax* itd.).

Slojevi višestrukog perceptrona su potpuno povezani (engl. *fully connected*), tj. svaki neuron iz jednog sloja je povezan sa svakim neuronom iz sljedećeg sloja, što je i vidljivo na slici 1.4.



Slika 1.4 Arhitektura višestrukog perceptrona [6]

1.2. Konvolucijske neuronske mreže

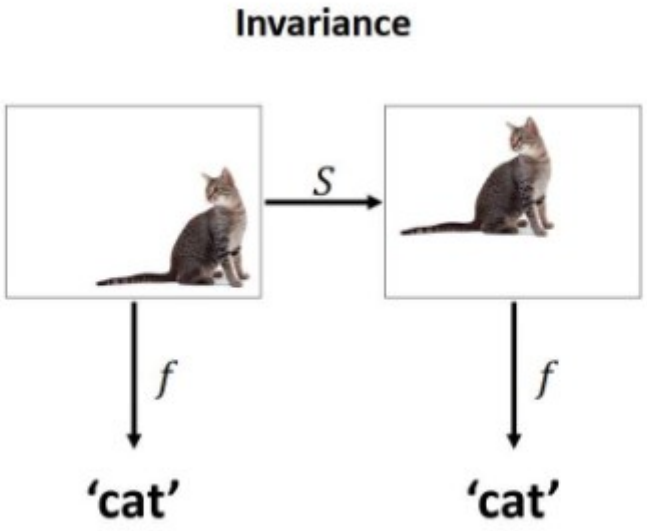
Konvolucijske neuronske mreže (engl. *convolutional neural networks*, CNNs) su specijalizirana vrsta umjetnih neuronskih mreža koja je prvenstveno namijenjena rješavanju problema iz domene prepoznavanja objekata, uključujući klasifikaciju, detekciju i segmentaciju slika.

Postoje tri velika razloga zbog kojih je potrebna posebna arhitektura za rješavanje ovakvih tipova problema [7]. Za početak, to je njihova računalna i memorijska složenost: tipična slika u problemima klasifikacije ima dimenzije 224×224, pri čemu za svaki piksel postoje tri vrijednosti (RGB). Ovo rezultira s ukupno 150.528 ulaznih vrijednosti (224×224×3). Ako se ovaj problem pokuša riješiti pomoću modela višeslojnog perceptrona sa samo jednim skrivenim slojem, taj model bi uz (uglavnom valjanu) pretpostavku da su

skriveni slojevi veći od ulaznih imao više od 22 milijardi težina. Ovo bi se definitivno htjelo izbjeći.

Drugi razlog je postojanje statističke veze između obližnjih piksela. Pikseli koji su blizu jedan drugome često predstavljaju dijelove istog objekta ili značajke što povlači povezanost između njihovih vrijednosti, ali višeslojni perceptron tu povezanost potpuno ignorira: svaka slika na ulazu u model se transformira u jednodimenzionalni vektor čime se gubi sva informacija o prostornoj strukturi slike.

Treći i zadnji razlog je taj što interpretacija slike ostaje ista čak i nakon provedbe geometrijskih transformacija nad njom. Na primjer, ako se nad slikom mačke obavi transformacija koja je prikazana na slici 1.5, to je i dalje slika mačke. Međutim, kada se ova transformirana slika preda na ulaz modela višeslojnog perceptrona, on ju vidi kao potpuno novu sliku, jer svaki piksel tretira kao zasebnu značajku. Kao rezultat toga, model višeslojnog perceptrona bi trebao naučiti uzorak piksela koji označava mačku na svakoj mogućoj poziciji, što je očito neefikasno.



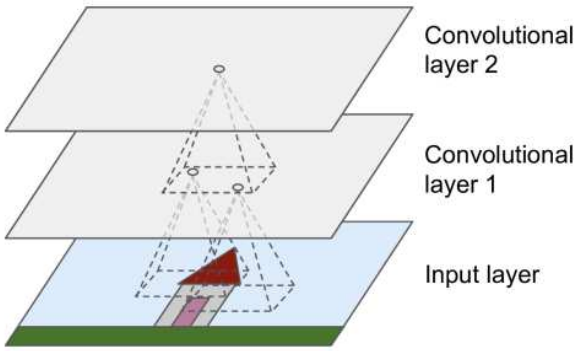
Slika 1.5 Slika mačke prije i nakon transformacije [8]

Konvolucijske neuronske mreže rješavaju ove probleme dodavanjem dodatnih slojeva u arhitekturu te će se o tim slojevima govoriti o nastavku.

1.2.1. Konvolucijski sloj

Konvolucijski sloj igra važnu ulogu u načinu na koji konvolucijske neuronske mreže funkcioniraju. Za razliku od slojeva višeslojnog perceptrona, gdje je svaki neuron u

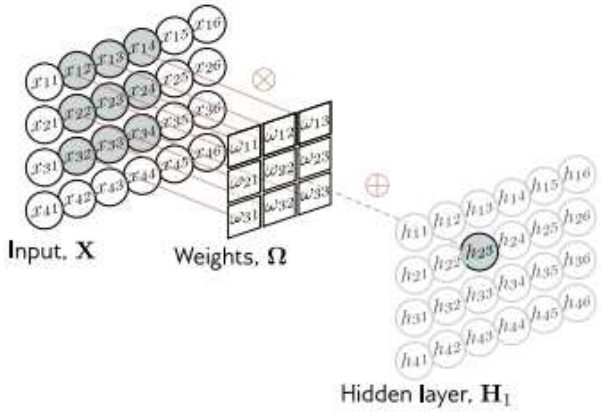
trenutnom sloju bio povezan sa svakim neuronom iz prethodnog, u konvolucijskom sloju svaki neuron je povezan samo s malom regijom neurona iz prethodnog sloja koja se naziva receptivnim poljem neurona (engl. *receptive field*). Slično kao što biološki neuroni vidne kore „vide“ samo određeni dio vidnog polja i reagiraju samo na podražaje koji dolaze iz njega [6], neuroni konvolucijskog sloja „vide“ samo one piksele obuhvaćene neuronima njihovog receptivnog polja i pri izračunu svoje izlazne vrijednosti uzimaju u obzir samo te neurone. Vizualizacija ovog koncepta je vidljiva na slici 1.6.



Slika 1.6 Konvolucijski slojevi sa prikazanim receptivnim poljima [6]

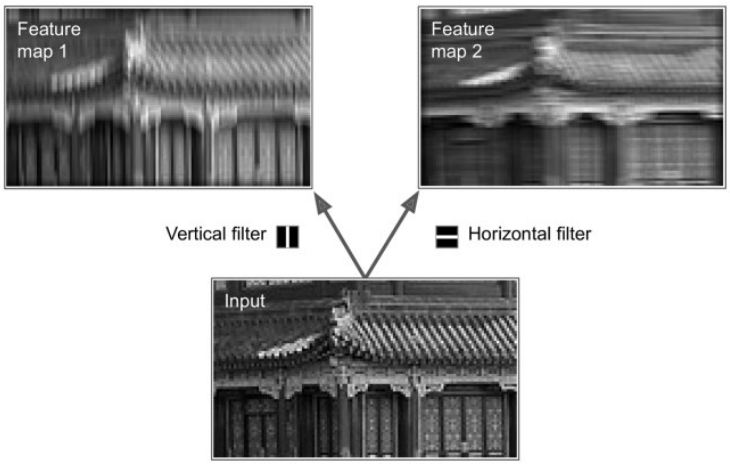
Konvolucijski slojevi se, kako i njihovo ime sugerira, temelje na konvoluciji. U kontekstu obrade slika, konvolucija predstavlja operaciju u kojoj se mala matrica koja se naziva filtrom pomiče preko ulazne slike te se pri svakom pomaku računa Hadamardov produkt između filtra i rešetke koja sadrži vrijednosti piksela dijela slike koji je trenutno pokriven filtrom. Rezultat produkta je matrica čije su dimenzije jednake dimenzijama ulaznih matrica, te se elementi ove dobivene matrice zatim sumiraju. Rezultat konvolucije je tzv. dvodimenzionalna aktivacijska mapa (engl. *activation map*), pri čemu svaki element ove mape predstavlja sumu za odgovarajući pomak filtra preko ulazne slike. Izlaz konvolucijskog sloja čini jedna ili više aktivacijskih mapa, po jedna za svaki filter koji sloj ima. Slika 1.7 prikazuje prethodno opisani proces.

Važno je napomenuti da ovi filtri nisu nužno dvodimenzionalni: u slučaju trodimenzionalnog ulaza (npr. slike u boji s tri RGB kanala ili izlaznih aktivacijskih mapa prethodnog konvolucijskog sloja) filtri i dalje „kližu“ po ulazu, ali „prožimaju“ cijelu njegovu dubinu, tj. i sami su trodimenzionalni.



Slika 1.7 Konstrukcija aktivacijske mape [7]

Vrijednosti filtra nisu slučajno odabrane: filtri su posebno dizajnirani tako da konvolucijom njih i ulazne slike rezultat bude aktivacijska mapa u kojoj su naglašene određene značajke, te se zbog toga aktivacijske mape često zovu i mapama značajki (engl. *feature maps*). Na slici 1.8 prikazane su dvije mape značajki nastale primjenom dva različita filtra na istu sliku: može se primijetiti da su na lijevoj slici naglašene vertikalne linije, dok je sve ostalo „zamagljeno“ i „nejasno“, a na desnoj je sličan slučaj, samo su tu naglašene horizontalne linije umjesto vertikalnih [6]. Upravo u ovome leži moć konvolucijskih neuronskih mreža: slaganjem konvolucijskih slojeva jedan na drugog, pri čemu niži slojevi prepoznaju jednostavnije značajke (npr. horizontalni i vertikalni rubovi), a viši slojevi pomoću nižih kompleksnije značajke (npr. kutove), omogućuje se prepoznavanje različitih obrazaca u slikama i time su dobre u rješavanju zadataka za koje su specijalizirane.



Slika 1.8 Mape značajki dobivene iz iste slike s dva različita filtra [6]

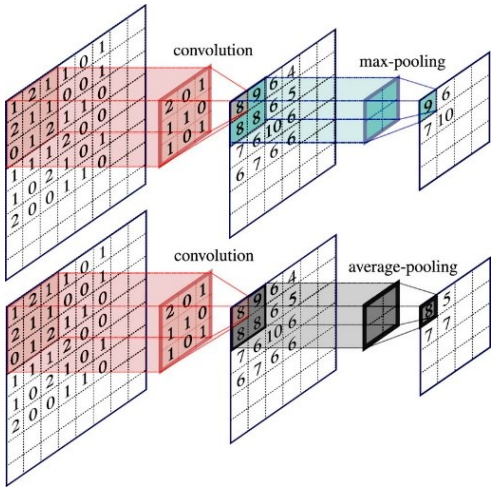
Iako su filtri na slici 1.8 odabrani ručno, u praksi kod učenja konvolucijskih neuronskih mreža se nikada ne koriste *apriori* filtri: taj zadatak prepušten je samoj neuronskoj mreži koja će pomoću algoritma propagacije pogreške unatrag sama korigirati vrijednosti filtara.

Uzimajući u obzir sve što je prethodno napisano, sada je jasno zašto se konvolucijski slojevi smatraju temeljnim građevnim blokom konvolucijskih neuronskih mreža: omogućuju analizu svakog malog dijela slike zasebno, smanjuju memorijsku složenost koristeći iste parametre na razini sloja, ne odbacuju statističku povezanost između vrijednosti susjednih piksela te rješavaju problem geometrijskih transformacija ulaza.

1.2.2. Sloj sažimanja

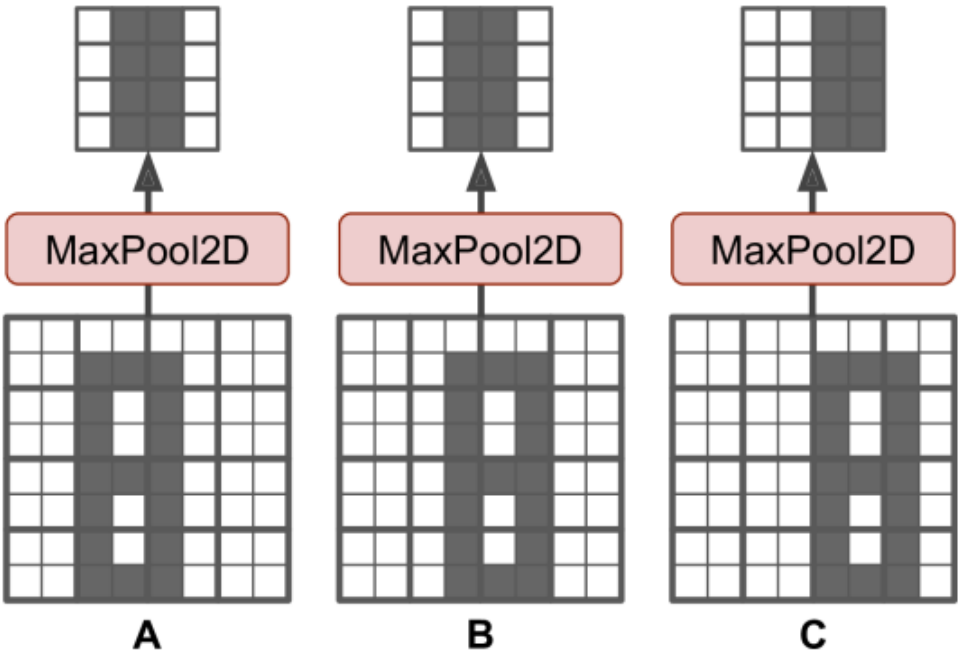
Sloj sažimanja (engl. *pooling layer*) je sloj čiji je primarni cilj smanjivanje dimenzija mapa značajki primljenih na ulazu, a time i daljnje smanjenje broja parametara i računalne složenosti modela.

Slično kao i kod konvolucijskog sloja, u sloju sažimanja također postoji filtar koji se pomiče po ulaznim mapama značajki, pri čemu se sve vrijednosti mape zahvaćene filtrom agregiraju u jednu vrijednost pomoću neke statističke obrade kao što je odabir najvećeg elementa ili izračun prosječne vrijednosti, što je i prikazano na slici 1.9. U praksi se najčešće koristi odabir najvećeg elementa, pri čemu su dimenzije filtra 2×2, a veličina koraka 2. Ovakav postupak svaku ulaznu mapu značajki transformira u mapu čija je veličina jednaka 25% veličine izvorne mape [9].



Slika 1.9 Operacije koje provode konvolucijski sloj i sloj sažimanja [10]

Sloj sažimanja je, osim što smanjuje broj parametara i računalnu složenost, također vrlo koristan pri očuvanju invarijantnosti prema malim translacijama piksela: to znači da, ako se ulazna slika translatira za mali iznos, izlaz sloja sažimanja se neće znatno promijeniti što je i demonstrirano na slici 1.10 [11]. To npr. može biti od posebne važnosti u rješavanju problema klasifikacije slika, gdje naučeni model mora u što većoj mjeri biti otporan na takve transformacije.



Slika 1.10 Invarijantnost izlaza sloja sažimanja za male translacije ulaza [6]

2. Konstrukcija i učenje modela neuronskih mreža

U ovom poglavlju je predstavljen podatkovni skup korišten u sklopu izrade modela klasifikacije slika cvijeća, proces predobrade ulaznih slika, sklopljene arhitekture višeslojnog perceptrona i konvolucijske neuronske mreže te rezultati njihovog učenja. Za konstrukciju i učenje prethodno spomenutih modela korišten je programski jezik Python te knjižnice TensorFlow i Keras koje olakšavaju rješavanje problema iz područja dubokog učenja.

2.1. Podatkovni skup

U sklopu učenja klasifikacijskog modela korišten je podatkovni skup *Oxford 102 Flower* čiji su tvorci Maria-Elena Nilsback i Andrew Zisserman sa Sveučilišta u Oxfordu. Skup se sastoji od ukupno 102 kategorije različitih cvjetova koji se često mogu naći u Ujedinjenom Kraljevstvu, pri čemu svaka kategorija sadrži od 40 do 258 različitih slika.

Između slika postoje varijacije u veličini, perspektivi i osvjetljenju. Dodatno, postoje kategorije čiji cvjetovi se mogu znatno razlikovati jedan od drugoga te nekoliko jako sličnih kategorija, što je i vidljivo na slici 2.1 [12].

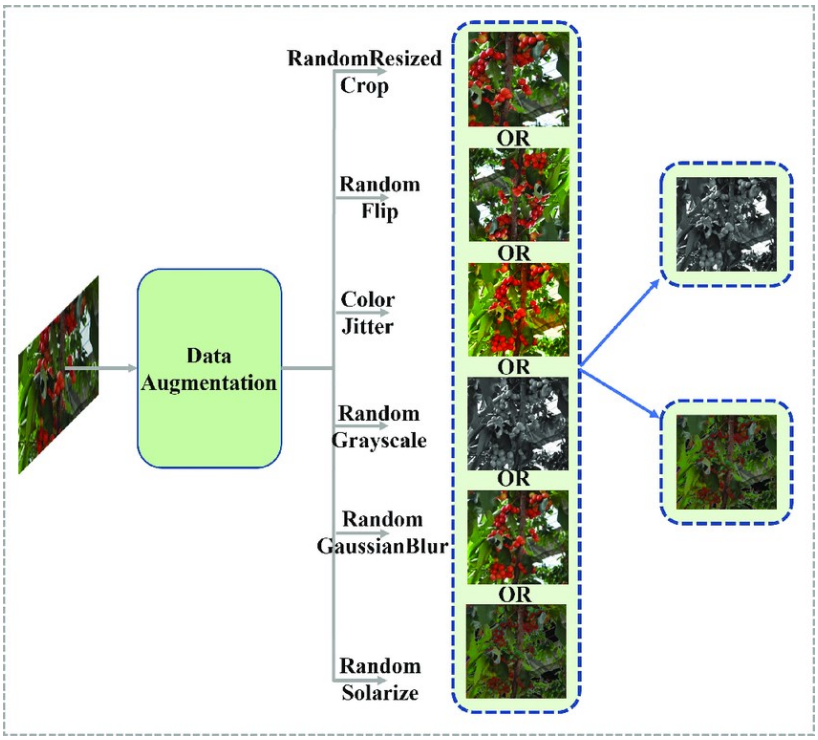
Slika 2.1 Sličnost različitih kategorija podatkovnog skupa [12]

Podatkovni skup ukupno sadrži 8189 slika, pri čemu njih 5778 čini skup za učenje (70,56%), 1185 skup za provjeru (14,47%) i 1226 skup za testiranje (14,97%). Skupovi su oblikovani tako da je iz svake kategorije 70% slika izdvojeno u skup za učenje, 15% u skup za provjeru i 15% u skup za testiranje.

2.2. Predobrada ulaznih podataka

Nakon što je skup podataka podijeljen na skupove za učenje, provjeru i testiranje, pojavljuje se određeni problem. Naime, iako skup za učenje sadrži skoro 6000 različitih slika, neke vrste cvjetova su poprilično slabo zastupljene: za pojedine kategorije u skupu za učenje postoji samo 28 različitih primjeraka, što je jako malo kad se uzme u obzir sveukupna veličina skupa. Ovaj manjak dovoljne količine podataka za učenje i neujednačena ravnoteža razreda unutar podatkovnog skupa je jedan od najčešćih problema strojnog učenja, a jedan od načina rješavanja ovog problema je tzv. augmentacija podataka [13].

U kontekstu klasifikacije slika, augmentacija podataka (engl. *data augmentation*) predstavlja tehniku kojom se povećava raznolikost i veličina skupa za učenje tako da se na postojeće slike primjenjuju slučajne, ali realistične transformacije (npr. obrezivanje, rotiranje, zumiranje i dr.). Primjer augmentacije slike prikazan je na slici 2.2.



Slika 2.2 Augmentacija slike [14]

Augmentacija slika se obavlja uz pomoć razreda ImageDataGenerator iz biblioteke Keras. Ovaj razred omogućuje specifikaciju slučajnih transformacija koje će se obaviti na slikama skupa za učenje na početku svake nove epohe učenja što znači da se model u svakoj epohi uči na „novim“ slikama. Kôd za konfiguraciju instance razreda ImageDataGenerator korištene u sklopu učenja prikazan je u nastavku:

```
training_data_preprocessor = ImageDataGenerator(
      rescale = 1. / 255,
      rotation_range = 20,
      width_shift_range = 0.1,
      height_shift_range = 0.1,
      shear_range = 0.2,
      zoom_range = 0.2,
      horizontal_flip = True
)
```

Kôd 2.1 Konfiguracija instance razreda ImageDataGenerator

Postavljanjem parametra rescale na vrijednost 1/255 zapravo se osigurava da se vrijednosti piksela slike prije ulaska u model podijele s vrijednošću 255, tj. vrijednosti piksela slike, koje se inače nalaze u intervalu [0, 255], sada se nalaze u intervalu [0, 1]. Ovaj postupak se naziva normalizacija podataka i u kontekstu rješavanja problema klasifikacije slika je iznimno važan jer gradijentni spust konvergira puno brže s normalizacijom podataka nego bez nje [15]. Ova vrsta normalizacije gdje se vrijednosti skaliraju u neki interval [a, b] se naziva normalizacija min-max i njezina općenita formula je prikazana na slici 2.3.

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)}$$

Slika 2.3 Normalizacija min-max [15]

Ostale transformacije koje mogu biti obavljene nad ulaznom slikom su njezino rotiranje za maksimalno 20° (parametar rotation_range), pomak slike vertikalno odnosno horizontalno za maksimalno 10% njezine visine odnosno širine (parametri height_shift_range i width_shift_range), smicanje slike za maksimalno 20% (parametar shear_range), zumiranje u ili iz slike za maksimalno 20% (parametar zoom_range) te horizontalno zrcaljenje (parametar horizontal_flip). S obzirom na to da slike nisu jednake veličine, svakoj slici su njezine dimenzije postavljene na 224×224.

2.3. Opis sklopljenih arhitektura

2.3.1. Arhitektura višeslojnog perceptrona

Kôd programske implementacije višeslojnog perceptrona dan je u nastavku:

```
model = Sequential([
      Flatten(),
      Dense(units = 512, activation = „relu“),
      BatchNormalization(),
      Dropout(0.5),
      Dense(units = 256, activation = „relu“),
      BatchNormalization(),
      Dropout(0.5),
      Dense(units = 102, activation = „softmax“)
])
```

Kôd 2.2 Implementacija višeslojnog perceptrona

Prvi sloj predstavljene arhitekture čini tzv. sloj spljoštavanja (engl. *flatten layer*) koji svaku ulaznu sliku pretvara u jednodimenzionalno polje vrijednosti. Nakon nje dolazi potpuno povezani skriveni sloj koji čini 512 neurona s prijenosnom funkcijom ReLU. Kako bi se smanjila vjerojatnost prenaučenosti i time poboljšale performanse modela na neviđenim podacima, poslije svakog skrivenog sloja postavljena su dva dodatna sloja: sloj normalizacije mini grupa i sloj isključivanja.

U kontekstu učenja s mini grupama, normalizacija mini grupa (engl. *batch normalization*) predstavlja tehniku transformacije ulaza tako da se ulaz prvo normira, a zatim skalira i pomiče. Ako ulazni vektor ima d komponenata, onda je normalizacija ulaza ekvivalentna normalizaciji svake njegove komponente j zasebno, pri čemu se prosječna vrijednost i standardna devijacija potrebne za normalizaciju komponente j računa na skupu vrijednosti koji čine komponente j svakog uzorka i trenutne mini grupe veličine N [16]. Nakon provedene normalizacije, svaka komponenta j dobivenog vektora se množi s komponentom j vektora skaliranja γ (engl. *scale vector*) te joj se dodaje komponenta j vektora pomaka β (engl. *offset vector*). Vrijednosti vektora γ i β se računaju pomoću postupka propagacije pogreške unatrag [4]. Postupak provedbe ovog algoritma prikazan je na slici 2.4.

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

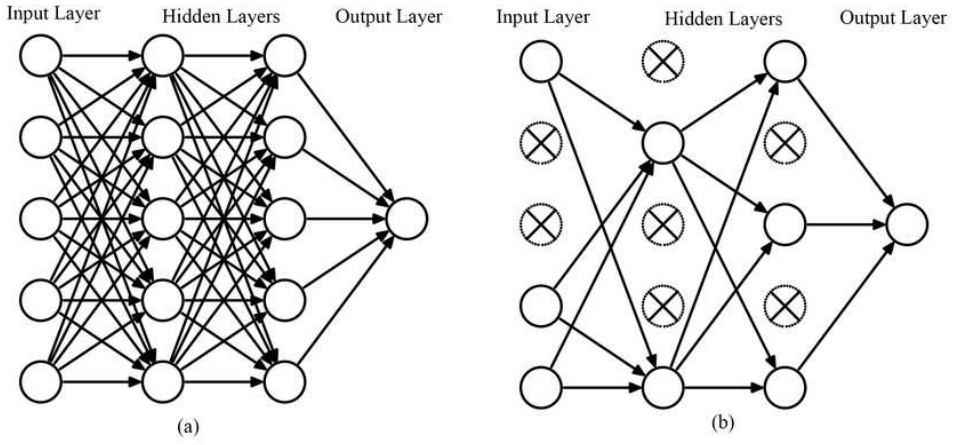
$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Slika 2.4 Algoritam normalizacije mini grupa [17]

Iako je normalizacija mini grupa primarno metoda umanjivanja internog kovarijantnog pomaka čime se sprječava mogućnost da gradijenti tijekom učenja poprimaju vrijednosti vrlo blizu nule i tako usporavaju proces učenja [11], ona je ujedno i regularizacijska tehnika, što znači da poboljšava sposobnost generalizacije modela.

Isključivanje (engl. *dropout*) je također metoda regularizacije, a funkcionira na sljedeći način: tijekom učenja se za svaki primjer iz skupa za učenje neuronska mreža modificira tako da izlaz svakog neurona iz ulaznog i skrivenih slojeva bude postavljen na nula s određenom vjerojatnošću. Ovime se osigurava da se neuroni ne oslanjaju u značajnoj mjeri na samo nekoliko ulaznih neurona, nego da obraćaju pažnju na svaki od njih što rezultira manjom osjetljivosti neurona na neznatne promjene ulaza pa time i robusnijom neuronskom mrežom [6]. Vizualizacija rezultata isključivanja neurona je vidljiva na slici 2.5.



Slika 2.5 Isključivanje neurona [18]

U ovoj arhitekturi hiperparametar slojeva isključivanja, tj. vjerojatnost da pojedini neuron sloja bude isključen, je postavljen na 0,5. Nakon prvog sloja isključivanja dolazi još jedan potpuno povezani sloj koji čini 256 neurona s prijenosnom funkcijom ReLU te zatim opet sloj normalizacije mini grupa i sloj isključivanja. U izlaznom sloju nalazi se 102 neurona koji za razliku od neurona prethodnih slojeva koriste prijenosnu funkciju *softmax*.

2.3.2. Arhitektura konvolucijske neuronske mreže

Kôd programske implementacije konvolucijske neuronske mreže dan je u nastavku:

```
model = Sequential([
      Conv2D(filters = 32, kernel_size = 3, activation =
'relu', padding = 'same', input_shape = (224, 224, 3)),
      BatchNormalization(),
      Conv2D(filters = 32, kernel_size = 3, activation =
'relu', padding = 'same'),
      BatchNormalization(),
      MaxPooling2D(pool_size = 2),
      Dropout(rate = 0.25),

      Conv2D(filters = 64, kernel_size = 3, activation =
'relu', padding = 'same'),
      BatchNormalization(),
      Conv2D(filters = 64, kernel_size = 3, activation =
'relu', padding = 'same'),
      BatchNormalization(),
      MaxPooling2D(pool_size = 2),
      Dropout(rate = 0.25),

      Conv2D(filters = 128, kernel_size = 3, activation =
'relu', padding = 'same'),
      BatchNormalization(),
      Conv2D(filters = 128, kernel_size = 3, activation =
'relu', padding = 'same'),
      BatchNormalization(),
      MaxPooling2D(pool_size = 2),
      Dropout(rate = 0.25),

      Flatten(),

      Dense(units = 512, activation = 'relu'),
```

```
        BatchNormalization(),
        Dropout(rate = 0.5),

        Dense(units = 102, activation = 'softmax')
])
```

Kôd 2.3 Implementacija konvolucijske neuronske mreže

Ulazne slike prvo prolaze kroz konvolucijski sloj u kojemu se konvolucijom njih i 32 različita filtra veličine 3×3 generiraju 32 mape značajki. S obzirom na to da je veličina ulaznih slika 224×224 i da se koristi nadopunjavanje (engl. *padding*), veličina izlaznih mapa je također 224×224. Iz istih razloga kao i kod arhitekture višeslojnog perceptrona i ovdje se koristi sloj normalizacije mini grupa. Poslije ova dva sloja opet se postavljaju ta dva ista sloja. Nakon što se ulaz propusti kroz prethodna četiri sloja dolazi do sloja sažimanja u kojemu se dimenzije svake ulazne mape značajki prepolove sa 224×224 na 112×112 pomoću filtra veličine 2×2. Postavljanjem sloja sažimanja smanjuje se broj parametara modela, a time i računalna složenost. Nakon sloja sažimanja dolazi sloj isključivanja s parametrom postavljenim na vrijednost 0,25 što znači da za svaki neuron sloja u svakom koraku učenja postoji šansa od 25% da bude isključen.

Cijela arhitektura se sastoji od više prethodno opisanih blokova slojeva, a okosnicu tih blokova čini upravo slaganje više konvolucijskih slojeva prije dodavanja sloja sažimanja. Ovakav način slaganja slojeva je česta pojava u arhitekturama konvolucijskih neuronskih mreža jer omogućuje ekstrakciju kompleksnijih značajki iz ulaznih slika [9].

Konačan rezultat prolaska ulazne slike kroz više ovakvih blokova čine 128 mapa značajki dimenzija 28×28. Sloj spljoštavanja pretvara dobivene mape značajki u jednodimenzionalno polje vrijednosti koje se zatim može proslijediti potpuno povezanom sloju od 512 neurona s prijenosnom funkcijom ReLU. Kao i kod arhitekture višeslojnog perceptrona, i u ovoj arhitekturi izlazni sloj čini 102 neurona sa prijenosnom funkcijom *softmax*.

2.4. Učenje modela

Kôd za učenje konstruiranih modela neuronskih mreža dan je u nastavku:

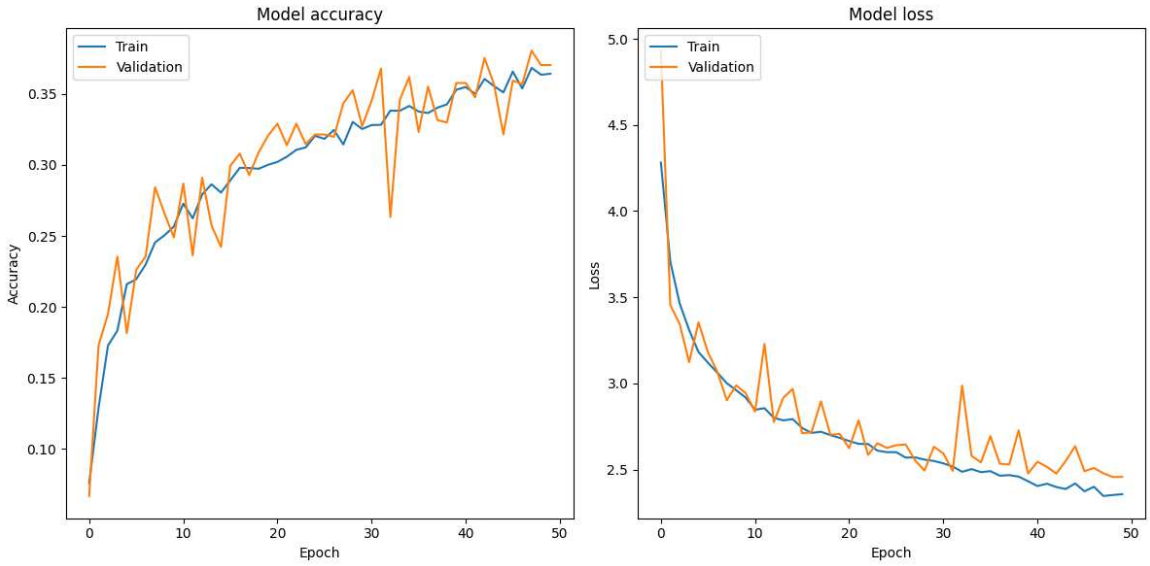
```
opt = Adam(learning_rate = 0.001)
model.compile(
      optimizer = opt,
      loss = 'sparse_categorical_crossentropy',
      metrics = ['accuracy']
)
early_stopping = EarlyStopping(
      monitor = 'val_loss',
      min_delta = 1e-3,
      patience = 10,
      verbose = 1,
      mode = 'auto',
      restore_best_weights = True
)
history = model.fit(
      train_generator,
      epochs = 50,
      steps_per_epoch = math.ceil(no_of_train_images /
batch_size),
      validation_data = validation_generator,
      validation_steps = math.ceil(no_of_validation_images) /
batch_size),
      callbacks = [early_stopping]
)
```

Kôd 2.4 Kôd za učenje modela

Umjesto klasičnog stohastičkog gradijentnog spusta za učenje modela koristi se adaptivna procjena momenta (engl. *adaptive moment estimation*, skraćeno *Adam*) koja spada u optimizacijske algoritme s adaptivnom stopom učenja te je vrlo često korištena zbog svoje efikasnosti i efektivnosti. Stopa učenja je postavljena na vrijednost 0,001, a za funkciju gubitka odabrana je kategorička unakrsna entropija (engl. *categorical crossentropy*). Model uči tijekom 50 epoha, ali ako u 10 epoha zaredom ne dođe do smanjenja gubitka na skupu za provjeru u iznosu od barem 0,001, model prestaje s učenjem te se vrate težine iz epohe u kojoj je model postigao najbolje performanse. Ovim potezom se smanjuje vjerojatnost prenaučenosti oblikovanih modela.

2.5. Rezultati učenja

Na slici 2.6 može se vidjeti vizualizacija procesa učenja modela višeslojnog perceptrona na razmatranom problemu.



Slika 2.6 Proces učenja modela višeslojnog perceptrona

Točnost i gubitak modela višeslojnog perceptrona na skupu za učenje stabilno rastu odnosno padaju, pri čemu svoj maksimum točnost postiže u 50. epohi sa vrijednošću 36,43%, a gubitak svoj minimum u 48. epohi sa vrijednošću 2,3465. Točnost i gubitak modela na skupu za provjeru također rastu odnosno padaju, ali to rade sa puno oscilacija, što bi mogao biti simptom prenaučenosti modela. Najveća preciznost modela na skupu za provjeru zabilježena je u 48. epohi i iznosi 38,06%, a najmanji gubitak je zabilježen u 49. epohi i iznosi 2,4569. Performanse modela na skupu za testiranje prikazane su na slici 2.7.

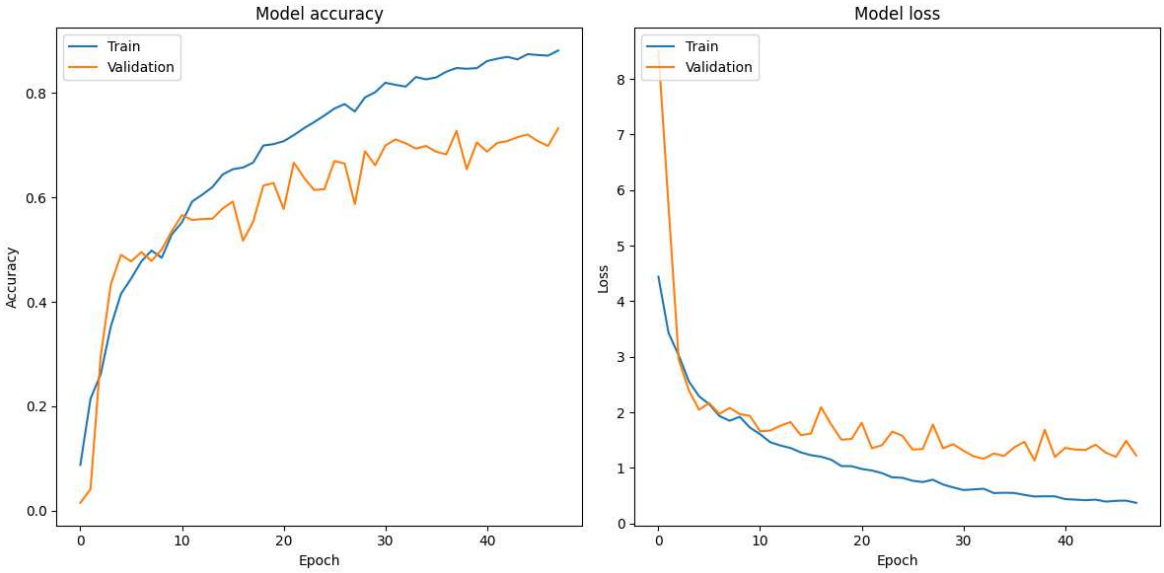
```
test_loss, test_acc = model.evaluate(test_generator, steps = math.ceil(1226 / batch_size))
print('Test accuracy:', test_acc)
```

✓ 3.9s

```
39/39 [==============================] - 4s 96ms/step - loss: 2.3769 - accuracy: 0.3915
Test accuracy: 0.39151713252067566
```

Slika 2.7 Performanse modela višeslojnog perceptrona na skupu za testiranje

Na slici 2.8 može se vidjeti vizualizacija procesa učenja modela konvolucijske neuronske mreže na razmatranom problemu.



Slika 2.8 Proces učenja modela konvolucijske neuronske mreže

Kako kod modela višeslojnog perceptrona, tako i kod modela konvolucijske neuronske mreže zabilježeni su konstantan rast točnosti i smanjenje pogreške na skupu za učenje, ali kod konvolucijskog modela su te krivulje dosta „snažnije“: u 48. epohi (koja je ujedno bila i zadnja epoha učenja jer je došlo do ranog zaustavljanja) ovaj model je imao točnost od čak 88,16% i gubitak u iznosu od 0,3728, što je značajno poboljšanje u odnosu na višeslojni perceptron. Međutim, ni ovdje nije moguće pobjeći oscilacijama u performansama na skupu za provjeru koje se kreću javljati u 17. epohi učenja. Najveću točnost na skupu za provjeru model konvolucijske neuronske mreže je imao u 48. epohi i iznosi 73,25%, a najmanji gubitak na skupu za provjeru je imao u 38. epohi, kada je ona iznosila 1,1335. S obzirom na to da u sljedećih 10 epoha nije došlo do minimalnog potrebnog smanjenja gubitka na skupu za provjeru, učenje prestaje u 48. epohi. Performanse modela na skupu za testiranje prikazane su na slici 2.9.

```
test_loss, test_acc = model.evaluate(test_generator, steps = math.ceil(1226 / batch_size))
print('Test accuracy:', test_acc)
```

```
39/39 [==============================] - 16s 415ms/step - loss: 1.2135 - accuracy: 0.7210
Test accuracy: 0.7210440635681152
```

Slika 2.9 Performanse modela konvolucijske neuronske mreže na skupu za testiranje

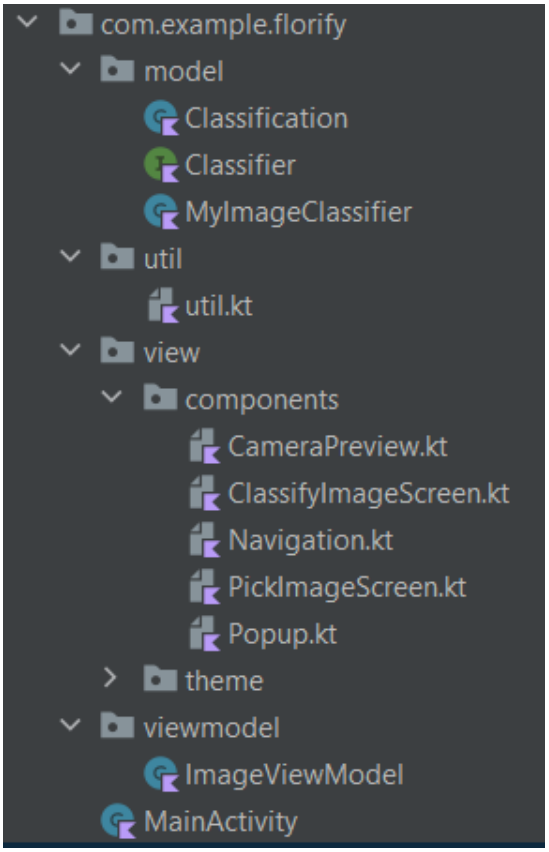
Model konvolucijske neuronske mreže očekivano postiže bolje rezultate kod rješavanja problema klasifikacije slika, pa je taj model ugrađen u mobilnu aplikaciju.

3. Implementacija mobilne aplikacije

U nastavku je predstavljena sama mobilna aplikacija, njezina struktura i najvažniji razredi te je prikazana njezina funkcionalnost na jednom ulaznom primjeru. Aplikacija je predviđena za operacijski sustav Android, a napisana je u programskom jeziku Kotlin uz korištenje Jetpack Composea, alata baziranog na Kotlinu koji kroz deklarativnu programsku paradigmu omogućuje olakšano pisanje i održavanje kôda korisničkog sučelja u aplikacijama za Android, te TensorFlow Litea, radnog okvira koji omogućuje korištenje modela strojnog učenja na uređajima s ograničenim resursima. Pri razvoju aplikacije korišteno je integrirano razvojno okruženje Android Studio.

3.1. Organizacija projekta

Na slici 3.1 prikazana je struktura aplikacije i raspodjela njezinih najvažnijih razreda po paketima.



Slika 3.1 Raspodjela razreda po paketima

3.1.1. Paket model

U ovom paketu se nalaze razredi i sučelja koji implementiraju samu poslovnu logiku aplikacije, tj. u ovom konkretnom slučaju klasifikaciju predanih slika cvijeća. Razred MyImageClassifier implementira sučelje Classifier i pruža uslugu klasifikacije slika koje mu se prosljeđuju kao instance razreda Bitmap. On funkcionira kao posrednik (engl. *proxy*): kada se nad njim pozove metoda za klasifikaciju, on taj poziv poslije izvođenja određene predobradne logike delegira servisu koji zna obaviti traženi zadatak, a to je instanca razreda ImageClassifier koji pripada knjižnici Task radnog okvira TensorFlow Lite. Kada servis izvede pozvanu metodu i vrati rezultat, MyImageClassifier iz dobivenog rezultata stvori niz objekata razreda Classification te klijentu vraća listu popunjenu tim objektima. Razred Classification je jednostavni podatkovni razred koji sadrži samo 2 člana: ime kategorije cvijeća (vrijednost tipa String) te pouzdanost modela da predana slika pripada upravo toj kategoriji (vrijednost tipa Float u intervalu [0, 1]).

3.1.2. Paket components

Paket components čine datoteke koje su primarno i isključivo vezane za oblikovanje korisničkog sučelja mobilne aplikacije: u datoteci PickImageScreen.kt nalazi se kôd za oblikovanje početnog ekrana, datoteka ClassifyImageScreen.kt sadržava kôd za oblikovanje ekrana zaduženog za prikaz rezultata klasifikacije, dok je u datoteci Popup.kt smješten kôd za oblikovanje skočnog prozora koji korisniku omogućuje da dobije predikciju klasifikacijskog modela za odabranu sliku ili, ako se u međuvremenu predomislio, odustane i uslika/odabere novu sliku. Svaka od ovih datoteka sadrži po jednu tzv. sastavljivu funkciju (engl. *composable function*) kojom je svako od gore navedenih ponašanja opisano. Sastavljive funkcije su Kotlinove funkcije s anotacijom @Composable kojima se na deklarativni način opisuje željeni izgled i ponašanje kako određenog dijela korisničkog sučelja: one predstavljaju osnovni građevni blok za izradu korisničkog sučelja.

3.1.3. Paket viewmodel

U ovom paketu se nalazi samo razred ImageViewModel. On nasljeđuje razred ViewModel te ima dva podatkovna člana: privatni podatkovni član tipa MutableStateFlow<Bitmap?> te javni podatkovni član tipa StateFlow<Bitmap?>. Instanca razreda MutableStateFlow<Bitmap?> funkcionira kao spremnik koji pohranjuje trenutno

odabranu sliku, a instanca razreda StateFlow<Bitmap?> predstavlja njegovu „read-only“ verziju koja funkcionira kao izdavač informacije o trenutno odabranoj slici te pri svakoj promjeni pohranjene slike obavještava o tome sve svoje pretplatnike, tj. sastavljive funkcije koje u skladu s tom promjenom ažuriraju dijelove korisničkog sučelja za koje su zadužene. Tako se npr. omogućuje prikaz skočnog prozora: kada se odabere slika, vrijednost pohranjena unutar spremnika se promjeni pa se o toj promjeni obavijesti sastavljiva funkcija u datoteci PickImageScreen.kt koja u skladu s time ažurira početni ekran. Ovakvo rješenje je slično onome koje sugerira oblikovni obrazac Promatrač, gdje bi instanca razreda StateFlow<Bitmap?> bila subjekt, a sastavljive funkcije konkretni promatrači. Osim ovoga, ImageViewModel također služi i za razmjenu podataka između ekrana.

3.1.4. Paket util

Ovaj paket također sadrži samo jednu datoteku util.kt, a u njoj se nalaze različite funkcije koje pomažu pri izvršavanju različitih zadataka (npr. funkcija za odabir slike iz galerije, funkcija koja provjerava ima li aplikacija sva potrebna dopuštenja i dr.).

3.2. Arhitektura aplikacije

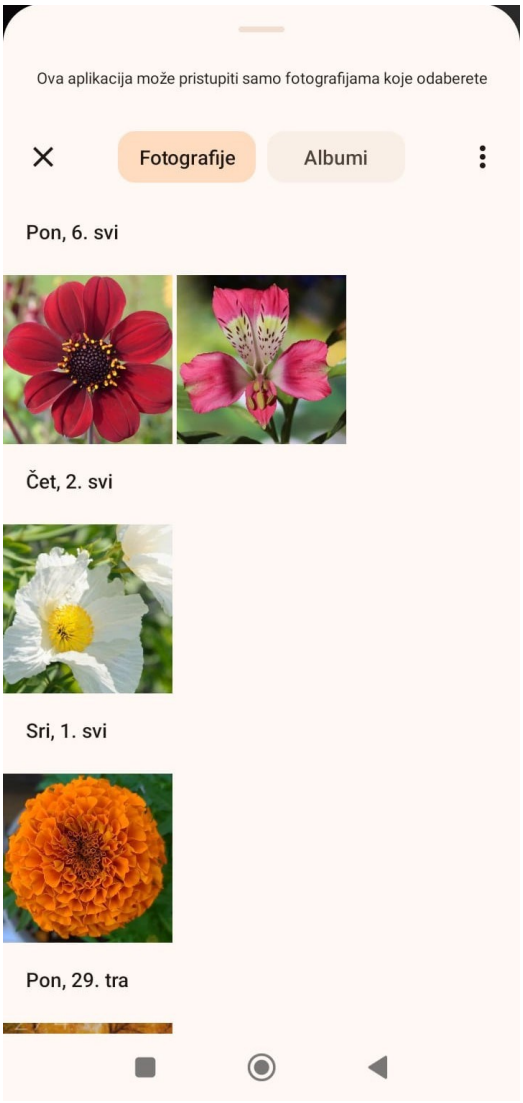
Ova aplikacija ima popularnu MVVM (engl. *Model-View-ViewModel*) arhitekturu, pri čemu Model čine razredi u paketu model, View čine sastavljive funkcije u datotekama paketa components, a ViewModel čini razred ImageViewModel iz paketa viewmodel. Grafički prikaz ovog arhitekturnog obrasca prikazan je na slici 3.2.

Slika 3.2 Arhitektura MVVM [19]

3.3. Rad aplikacije

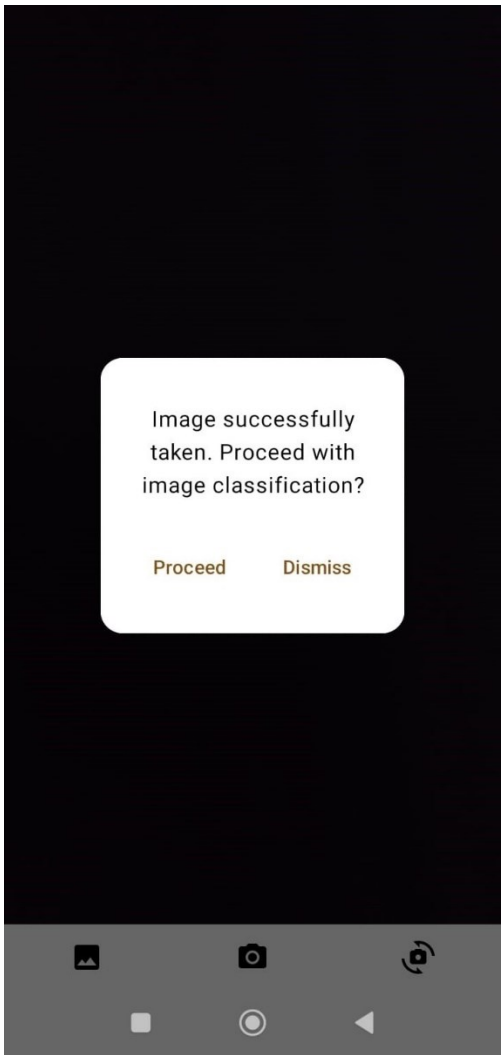
Postoje dva različita načina na koja je moguće unijeti sliku cvijeta u aplikaciju: moguće je uslikati cvijet direktno unutar aplikacije koristeći funkcionalnosti koje nudi knjižnica CameraX, ili se može odabrati slika cvijeta iz galerije. Izgled korisničkog sučelja aplikacije u slučaju odabira prvog načina je prikazan na slici 3.3, a u slučaju drugog načina na slici 3.4.

Slika 3.3 Izgled korisničkog sučelja prilikom slikanja cvijeta



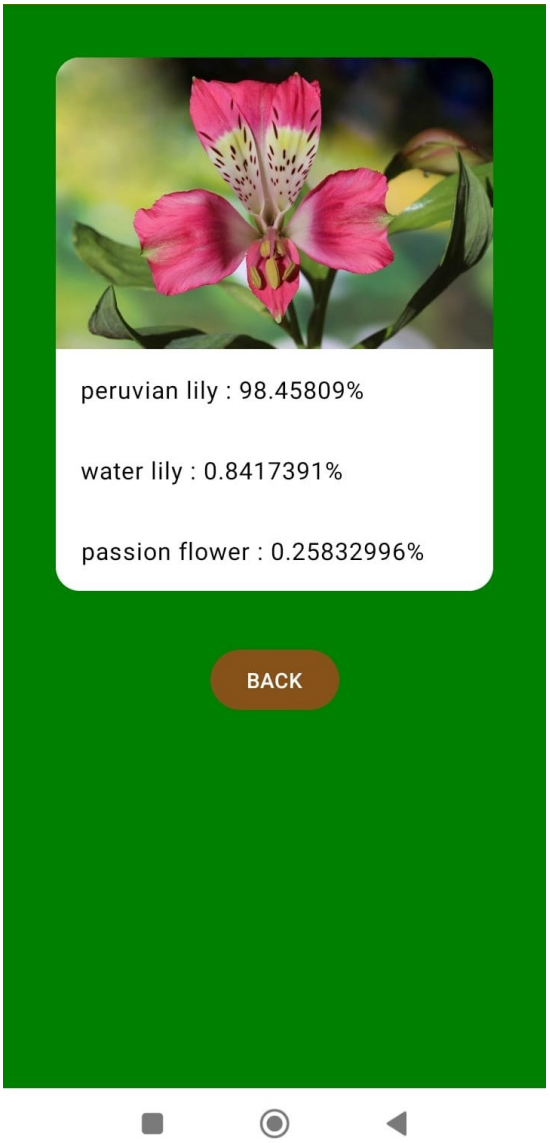
Slika 3.4 Izgled korisničkog sučelja prilikom odabira slike cvijeta iz galerije

Nakon što korisnik uslika ili odabere sliku iz galerije, na ekranu će se pojaviti prozor koji korisniku nudi dvije opcije: može ili odabrati opciju „Proceed“, pri čemu se pohranjena slika šalje na predobradu i zatim prosljeđuje modelu dubokog učenja koji će analizirati odabranu sliku i identificirati vrstu cvijeta, ili odabrati opciju „Dismiss“ što omogućuje korisniku da ponovno uslika/odabere željenu sliku. Sama predobrada i klasifikacija slike traju dovoljno dugo da pokvari korisničko iskustvo, pa je ova opcija dodana kako korisnik npr. u slučaju odabira krive slike iz galerije ne treba čekati da se proces završi. Izgled ekrana u ovom stanju prikazan je na slici 3.5.



Slika 3.5 Izgled početnog ekrana sa skočnim prozorom

Ako korisnik odabere opciju „Proceed“, slika se šalje modelu koji kao rezultat vraća tri najvjerojatnije predikcije s pripadnim vjerojatnostima koje govore koliko je model siguran u pojedinu predikciju. Prikaz ekrana koji prikazuje rezultate klasifikacije vidljiv je na slici 3.6.



Slika 3.6 Izgled ekrana koji prikazuje rezultate klasifikacije

Zaključak

Aplikacija za automatiziranu klasifikaciju cvijeća je uspješno implementirana: koristi model temeljen na konvolucijskim neuronskim mrežama koji na skupu za testiranje daje točnost od 72,10% kako bi klasificirala predanu sliku. Iako je aplikacija funkcionalna, definitivno ima mjesta za poboljšanje. Za početak, korišteni model je učen na podatkovnom skupu koji sadrži slike samo onih cvjetova koji se mogu naći u Ujedinjenom Kraljevstvu. Ovakva aplikacija definitivno nije od pomoći korisniku koji živi u Indiji ili Mauricijusu te bi se konstruirani model trebao učiti na većem podatkovnom skupu koji sadrži cvjetove iz različitih krajeva svijeta. Osim toga, također bi trebalo poboljšati samu arhitekturu korištenog modela. Iako točnost od 72,10% nije loša, i dalje je daleko od rezultata koje postižu „state-of-the-art“ modeli. I za kraj, također bi bilo korisno da se osim samih predikcija modela ispišu i neke osnovne biološke zanimljivost o razredu za koji model ima najveću sigurnost, kako bi korisnik naučio nešto novo.

Literatura

[1] Andina, D., Pham, D. T. *Computational intelligence: For engineering and manufactoring*. 1. izdanje. New York: Springer, 2007.

[2] *Neuronska mreža*, Hrvatska enciklopedija. Poveznica: https://www.enciklopedija.hr/clanak/neuronska-mreza; pristupljeno 7. svibnja 2024.

[3] Ujević Andrijić, Ž. *Umjetne neuronske mreže*, Osvježimo znanje, 68, 2019, str. 219

[4] Wang, X., Liu, Y., Xin, H. *Bond strength prediction of concrete-encased steel structures using hybrid machine learning method*, Structures, 32, 2021, str. 2285

[5] *What is ReLU and Sigmoid activation function?* Poveznica: https://www.nomidl.com/deep-learning/what-is-relu-and-sigmoid-activation-function/; pristupljeno 3. lipnja 2024.

[6] Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow*, 2. izdanje. O'Reilly Media, Inc., 2019

[7] Prince, S. J. D. *Understanding Deep Learning*. The MIT Press, 2023

[8] *05 Imperial's Deep learning course: Equivariance and Invariance.* Poveznica: https://www.youtube.com/watch?v=a4Quhf9NhMY&t=944s; pristupljeno 12. svibnja 2024.

[9] O'Shea, K., Nash, R. *An Introduction to Convolutional Neural Networks*, 2015

[10] Teo, Y. S., Shin, S., Jeong, H., Kim, Y., Kim, Y. H., Struchalin, G. I., Kovlakov, E. V., Straupe, S. S., Kulik, S. P., Leuchs, G., Sánchez-Soto, L. L. *Benchmarking quantum tomography completeness and fidelity with machine learning*, New Journal of Physics, 23, 2021

[11] Džomba, K. *Konvolucijske neuronske mreže*. Diplomski rad. Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, 2018

[12] *Oxford 102 Flower (102 Category Flower Dataset).* Poveznica: https://paperswithcode.com/dataset/oxford-102-flower; pristupljeno 5. svibnja 2024.

[13] Mikołajczyk, A., Grochowski, M. *Data augmentation for improving deep learning in image classification problem*. International Interdisciplinary PhD Workshop, Swinoujscie, (2018)

[14] Gai, R.-L., Wei, K., Wang, P.-F. *SSMDA: Self-Supervised Cherry Maturity Detection Algorithm Based on Multi-Feature Contrastive Learning*, Agriculture, 13, 2023

[15] *Feature scaling.* Poveznica: https://en.wikipedia.org/wiki/Feature_scaling; pristupljeno 25. svibnja 2024.

[16] *Batch normalization*. Poveznica: https://en.wikipedia.org/wiki/Batch_normalization; pristupljeno 30. svibnja 2024.

[17] Li, F.-F., Johnson, J., Yeung, S. *Lecture 7: Training Neural Networks, Part 2.* Sveučilište Stanford, 2018. Poveznica: https://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture07.pdf; pristupljeno 30. svibnja 2024.

[18] Wang, Z.-S., Lee, J., Song, C. G., Kim, S.-J. *Efficient Chaotic Imperialist Competitive Algorithm with Dropout Strategy for Global Optimization*, Symmetry, 12, 2020

[19] *MVVM architecture.* Poveznica: https://commons.wikimedia.org/wiki/File:MVVM_architecture.svg; pristupljeno 1. lipnja 2024.

Sažetak

Mobilna aplikacija za klasifikaciju vrste cvijeća sa slike korištenjem metoda dubokog učenja

U ovom završnom radu predstavljena je mobilna aplikacija koja omogućuje klasifikaciju slika cvijeća pomoću ugrađenog modela dubokog učenja. U sklopu učenja korišten je podatkovni skup *Oxford 102 Flower* koji sadrži 102 kategorije različitih cvjetova koji se često mogu naći u Ujedinjenom Kraljevstvu. Za razvoj modela dubokog učenja korištena su dvije arhitekture: višeslojni perceptron i konvolucijska neuronska mreža. Model zasnovan na arhitekturi višeslojnog perceptrona je na skupu za testiranje imao točnost od 39,15%, dok je model zasnovan na arhitekturi konvolucijske neuronske mreže imao točnost od 72,10% pa je on ugrađen u aplikaciju. Sama mobilna aplikacija je napisana u programskom jeziku Kotlin te je namijenjena za operacijski sustav Android.

Ključne riječi: klasifikacija; duboko učenje; konvolucijska neuronska mreža; višeslojni perceptron; Keras; TensorFlow; Android; Kotlin

Summary

Mobile application for flower species classification from an image using deep learning methods

In this bachelor thesis, a mobile application that enables flower image classification using an embedded deep learning model is presented. The *Oxford 102 Flower* dataset, which contains 102 categories of different flowers commonly found in the United Kingdom, was used for training. Two architectures were used for the development of the deep learning model: a multilayer perceptron and a convolutional neural network. The model based on the multilayer perceptron architecture achieved an accuracy of 39.15% on the test set, while the model based on the convolutional neural network architecture achieved an accuracy of 72.10%, and thus it was integrated into the application. The mobile application itself is written in the Kotlin programming language and is intended for the Android operating system.

Keywords: classification; deep learning; convolutional neural network; multilayer perceptron; Keras; TensorFlow; Android; Kotlin