SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 163

# MOBILNA APLIKACIJA ZA KLASIFIKACIJU KOŽNIH OBOLJENJA IZ SLIKA KOŽE

Petar Miličević

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 163

# MOBILNA APLIKACIJA ZA KLASIFIKACIJU KOŽNIH OBOLJENJA IZ SLIKA KOŽE

Petar Miličević

Zagreb, lipanj 2021.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 12. ožujka 2021.

# ZAVRŠNI ZADATAK br. 163

| | |
|---|---|
| Pristupnik: | **Petar Miličević (0036515023)** |
| Studij: | Elektrotehnika i informacijska tehnologija i Računarstvo |
| Modul: | Računarstvo |
| Mentor: | izv. prof. dr. sc. Alan Jović |

Zadatak: **Mobilna aplikacija za klasifikaciju kožnih oboljenja iz slika kože**

Opis zadatka:

Cilj završnog rada je izrada mobilne aplikacije koja će omogućiti klasifikaciju kožnih oboljenja na temelju fotografija kože. Aplikacija treba biti izrađena za operacijski sustav Android i omogućiti fotografiranje kože, prijenos fotografije na poslužiteljsku stranu te ispis rezultata klasifikacije. Klasifikacija na poslužiteljskoj strani treba se temeljiti na modelu duboke neuronske mreže, koju je potrebno naučiti i optimirati na otvorenom skupu podataka o kožnim oboljenjima. Primjer takvog skupa podataka je dostupan na web stranici: https://challenge2018.isic-archive.com/. Rezultate klasifikacijskog algoritma potrebno je usporediti sa sličnim rješenjima. Aplikacija treba omogućiti registraciju korisnika, prijavu, te pohranu podataka o korisniku i klasifikacije slika u bazi podataka. Izbor programskog jezika i alata za ostvarenje ovog sustava je proizvoljan.

Rok za predaju rada: 11. lipnja 2021.

# SADRŽAJ

**1. Uvod** **1**

**2. Koža i kožne bolesti** **3**

2.1. Koža . . . . . 3

2.2. Kožne bolesti . . . . . 5

2.3. Umjetna inteligencija u medicini . . . . . 7

**3. Duboke neuronske mreže** **8**

3.1. Duboko učenje i neuronske mreže . . . . . 8

3.2. Konvolucijske neuronske mreže . . . . . 13

**4. Izgradnja modela duboke neuronske mreže** **21**

4.1. Korištene tehnologije i alati . . . . . 21

4.2. Podaci . . . . . 22

4.3. Priprema podataka . . . . . 24

4.4. Gradnja modela . . . . . 28

4.5. Učenje i testiranje modela . . . . . 29

4.6. Usporedba sa sličnim rješenjima . . . . . 33

**5. Izgradnja aplikacije** **34**

5.1. Opis sustava . . . . . 34

5.2. Arhitektura sustava . . . . . 34

5.3. Korištene tehnologije i alati . . . . . 36

5.4. Klijentska strana . . . . . 36

5.5. Poslužiteljska strana . . . . . 40

5.6. Baza podataka . . . . . 42

**6. Zaključak** **43**

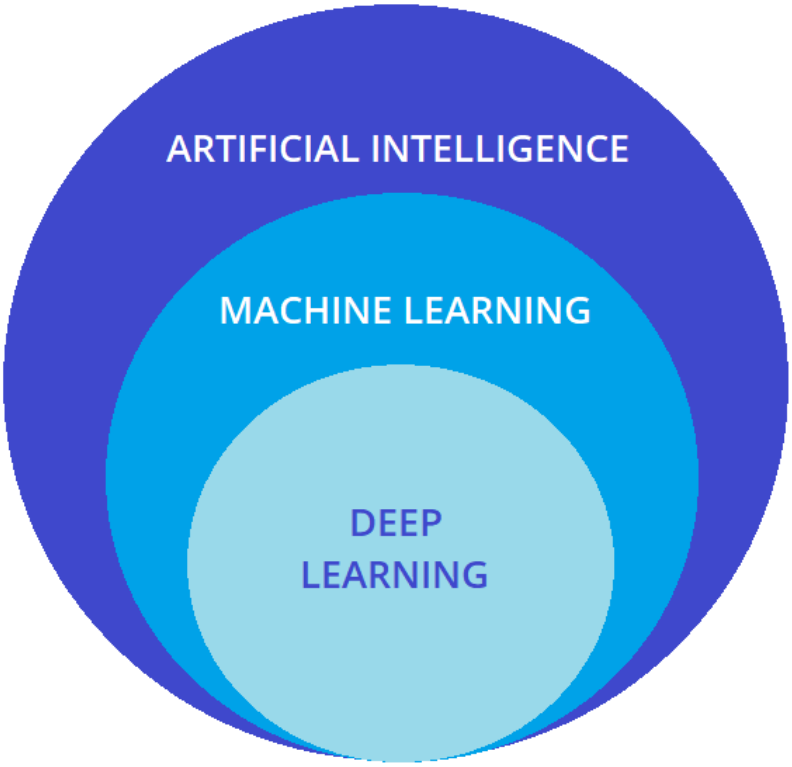
**7. literatura** **44**

# 1. Uvod

Strojno učenje jedna je od najbrže rastućih grana umjetne inteligencije. Glavni aspekt strojnog učenja koji ga odvaja od ostalih tehnika u umjetnoj inteligenciji je sposobnost prilagodbe i automatskog poboljšavanja algoritama kada su izloženi novim skupovima podataka. Zbog toga se smatra vrlo dinamičnim područjem koje ne ovisi toliko o ljudskim resursima kako bi se došlo do poboljšanja i novih saznanja [8].

Algoritmi strojnog učenja grade model zasnovan na ulaznom skupu podataka (engl. *training data*). Pomoću izgrađenog modela algoritmi potom mogu donositi samostalne odluke i predviđanja bez da su za to eksplicitno programirani. U zadnjem desetljeću algoritmi strojnog učenja omogućili su razvoj autonomne vožnje, učinkovitije pretraživanje sadržaja na webu i automatsko prepoznavanje ljudskog govora, kao i mnogih drugih područja i primjena [9].

Duboko učenje jedna je od tehnika strojnog učenja. Algoritmi dubokog učenja zasnivaju se na modelima dubokih neuronskih mreža. Duboke neuronske mreže su skup algoritama koji postižu najveće postotke točnosti za razne važne probleme, kao što su prepoznavanje slika i zvuka. Termin "duboke" odnosi se na broj slojeva u neuronskoj mreži, tako postoje mreže s jednim i više tzv. "skrivenih slojeva" (engl. *hidden layer*). Više takvih skrivenih slojeva omogućava dubokim neuronskim mrežama da na temeljitiji način nauče podatkovne značajke i njihovu hijerarhiju. Neke jednostavnije značajke rekombiniraju se iz jednog sloja u sljedeći pa je omogućeno stvaranje složenijih značajki, što rezultira dubljim poznavanjem podatkovnih međuovisnosti i obrazaca. Mreže s mnogo slojeva provode ulazni skup podataka kroz više skrivenih slojeva pa su stoga nerijetko vremenski zahtjevne za učenje [8].




**Slika 1.1:** Hijerarija područja umjetne inteligencije

Navedena područja imaju poseban značaj u razvoju medicine, njihova hijerarhija prikazana je na slici 1.1. Zahvaljujući postupcima strojnog učenja, liječnici mogu davati preciznije dijagnoze pacijentima, prognozirati daljnji razvoj nekih bolesti i predlagati prilagođenije načine liječenja [10]. Ovaj rad razmatra razvoj cjelokupnog rješenja, mobilne aplikacije koja omogućava klasifikaciju kožnih oboljenja na temelju slika kože.

# 2. Koža i kožne bolesti

Koža je najveći i najteži ljudski organ jer ima površinu od $1.2$ do $2.3$ $m^2$, a masa joj čini od 10 do 15 posto ukupne tjelesne mase [11].

S druge strane, rak kože najčešći je zloćudni tumor kod čovjeka. Povećanjem brzine rasta svoje incidencije u općoj populaciji, zadnja tri desetljeća izuzetno je izražen javnozdravstveni problem s kojim se bave gotovo sve zemlje u svijetu, pa tako i Republika Hrvatska. Prema posljednjim dostupnim informacijama Registra za rak Hrvatskog zavoda za javno zdravstvo, u 2014. godini bilo je ukupno 587 novooboljelih od melanoma, od toga 305 muškaraca i 282 žene. U dobnoj skupini od 30 do 39 godina jedan je od tri najčešća zloćudna tumora, uz karcinom testisa i štitnjače [11].

U većini slučajeva dijagnosticira se vizualno, počevši s početnim dermatološkim pregledom. Takav pristup otvara mogućnost uporabe algoritama dubokog učenja za pomoć u klasifikaciji oboljenja. Automatska klasifikacija kožnih oboljenja korištenjem slika izazovan je zadatak zbog velike varijabilnosti i nepravilnosti njihova izgleda.

## 2.1. Koža

Kao što je već spomenuto, koža je najveći ljudski organ. S kosom, noktima, žlijezdama i živcima dio je čovjekovog zaštitnog sustava koji djeluje kao prepreka između vanjske i unutarnje strane tijela. Debljina kože razlikuje se ovisno o dijelu tijela, najdeblja je na tabanima, oko $4$ $mm$, a najtanja na vjeđama, oko $0.2$ $mm$ [11]. Koža se sastoji od tri sloja tkiva: epidermisa, dermisa i hipodermisa, prikazanih na slici 2.1.

**Epidermis**

Epidermis je vanjski, vidljivi dio kože. Stanice epidermisa svakodnevno se obnavljaju jer se mrtve stanice neprestano odvajaju od ostatka tkiva. Glavne funkcije epidermisa su:

- Izrada novih kožnih stanica
- Davanje boje koži
- Zaštita kože

**Dermis**

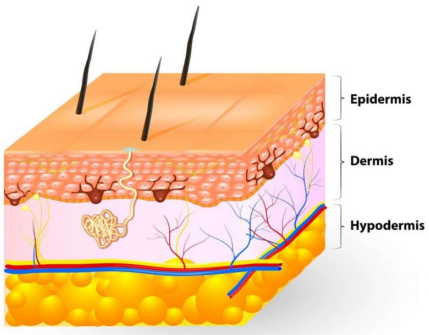
Dermis je srednji sloj kože koji se nalazi ispod epidermisa. To je najdeblji sloj kože koji sadrži živce i krvne žile. U njemu se također nalaze znojne i uljne žlijezde, kao i korijeni dlaka. Sastoji se uglavnom od proteina kolagena koji kožu čini rastezljivom i snažnom. Glavne funkcije dermisa su:

- Osjet boli i dodira
- Proizvodnja ulja i znoja
- Rast dlaka i kose
- Kontrola temperature kože
- Borba protiv infekcije

**Hipodermis**

Hipodermis je najdublji sloj kože, najčešće se koristi kao sinonim za potkožno masno tkivo. Glavne funkcije hipodermisa su:

- Izolacija tijela od topline i hladnoće
- Skladište energije
- Zaštita unutarnjih organa i kostiju [11]




**Slika 2.1:** Slojevi kože [1]

## 2.2. Kožne bolesti

Rak kože glavni je javnozdravstveni problem u svijetu, s više od $123000$ novih slučajeva svake godine. Najčešći je oblik karcinoma, koji čini oko $40$ posto ukupnog broja slučajeva raka [11]. Iako općenito postoji velik broj kožnih bolesti, u ovom radu fokus će biti na najčešće, tj. one koje su prisutne u skupu podataka koji će biti korišten za učenje i optimizaciju modela duboke neuronske mreže za klasifikaciju. Takav konkretan skup podataka je $HAM10000\ dataset$ [12]. Skup podataka $HAM10000$ sastoji se od $10015$ slika kožnih oboljenja koje su objavljene kao skup ulaznih podataka za potrebe razvoja akademskog strojnog učenja. Skup podataka sadrži $7$ različitih klasa kožnih oboljenja koja su navedena u nastavku:

1. Melanocytic nevi – Melanocitni nevusi (madeži) su dobroćudne (benigne) pigmentirane promjene građene od nevusnih stanica, melanocita. Obično se pojavljuju na koži i sluznicama, a prosječan broj melanocitnih nevusa kod odrasle osobe je od $20$ do $40$.

2. Melanoma – Melanom je zloćudni tumor koji nastaje iz melanocita-specijaliziranih stanica koje proizvode melanin, a nalaze se u dubokom sloju epidermisa. Melanin je prirodni kožni pigment koji daje boju koži, kosi i očima. Ako se otkrije u ranoj fazi, može se izliječiti jednostavnim kirurškim zahvatom, melanomi također mogu biti invazivni i neinvazivni (in situ).

3. Benign keratosis-like lesions – Benigna (seboroična) keratoza jedna je od najčešćih nekanceroznih izraslina na koži u starijih osoba. Obično se pojavljuje kao smeđa ili crna izraslina na licu, prsima, ramenima ili leđima. Često se pojavljuje u skupinama, a općenito je bezbolna i ne zahtijeva liječenje.

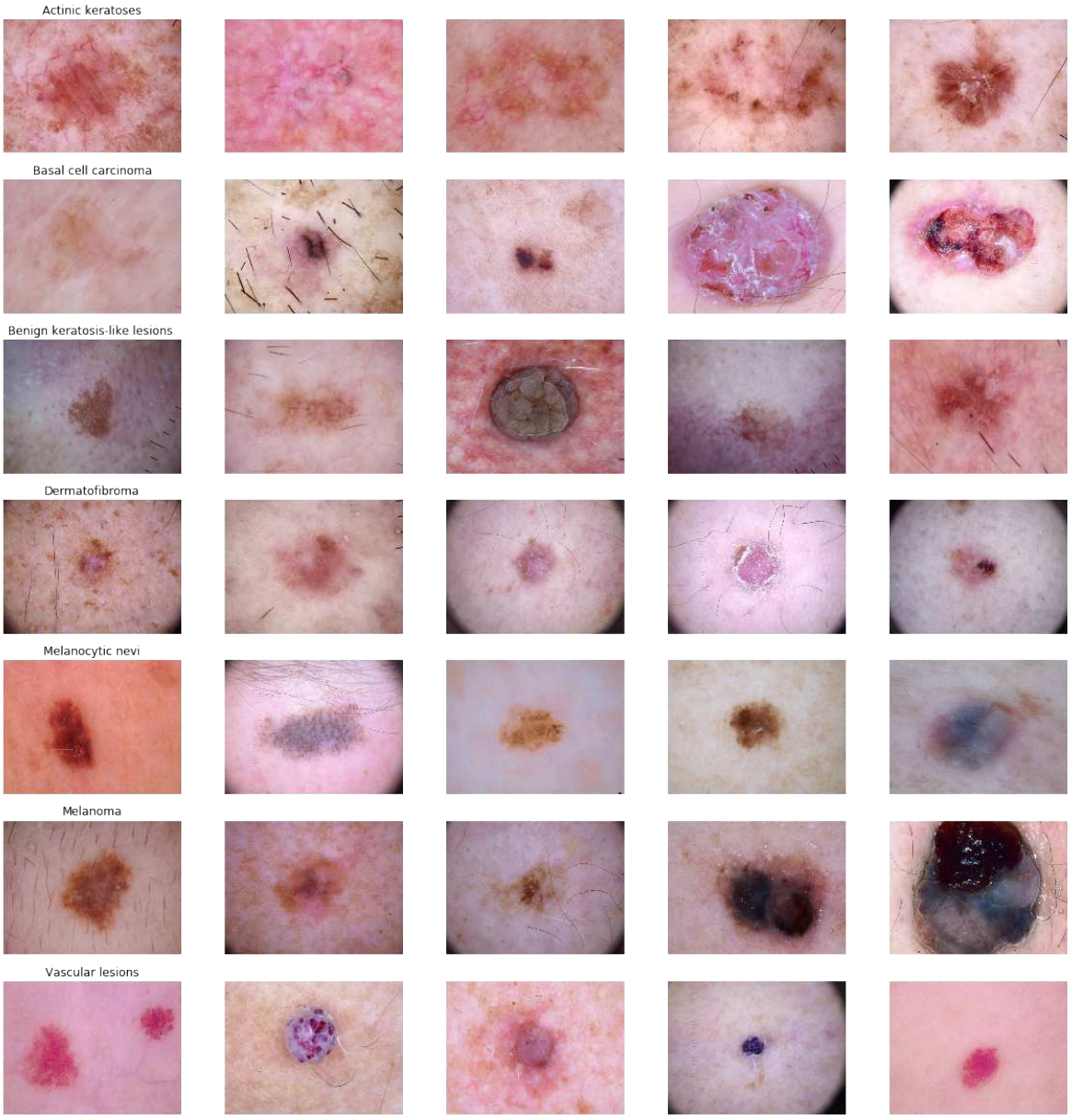
4. Basal cell carcinoma – Karcinom bazalnih stanica najčešća je vrsta kožnog karcinoma. Tumori započinju kao male čvrste uzdignute izrasline na koži ili rane nalik krastama. Rijetko stvara metastaze, ali destruktivno raste ako se ne liječi.

5. Actinic keratoses – Aktinična keratoza očituje se kao keratonična tvorba na dijelovima kože koji su izloženi ultraljubičastom (UV) zračenju sunca. Može prijeći u zloćudni (maligni) planocelularni karcinom (rak pločastih stanica). Jedna je od najčešćih kožnih bolesti.

6. Vascular lesions – Vaskularne lezije obuhvaćaju većinski bezopasne kožne izrasline. Ovdje spadaju angiom trešnje (engl. *cherry angiomas*), angiokeratom (engl. *anigokeratomas*) i pilogeni granulom (engl. *pyogenic granulomas*).

7. Dermatofibroma – Fibrom ili dermatofibrom je benigna kožna izraslina građena od vlaknastog i vezivnog tkiva. Uglavnom je boje kože, no može biti i pigmentiran. Ne zahtijeva liječenje [11].

Na slici 2.2 za svaku od navedenih klasa kožnih oboljenja prikazano je 5 primjera njihovog fizičkog izgleda na koži.




**Slika 2.2:** Primjeri kožnih oboljenja

## 2.3. Umjetna inteligencija u medicini

Kao što je već spomenuto, kožni karcinomi spadaju u najčešće karcinome općenito. Prema američkoj zakladi za rak kože (US Skin Cancer Foundation), u SAD-u se ta bolest godišnje dijagnosticira više nego svi ostali karcinomi zajedno [10]. Karcinomi se obično klasificiraju kao melanomski ili nemelanomski. Teško ih je razlikovati od uobičajenih benignih kožnih lezija, a fizički izgled posebno je varijabilan. Događa se da se:

- Rak kože zanemari jer se smatra bezopasnim
- Velik broj bezopasnih lezija odstranjuje u strahu od raka kože

Dermatolozi pregledavaju kožne lezije vizualno i dermatoskopijom. Oni se koriste iskustvom u prepoznavanju uzoraka kako bi odredili koji poremećaji su potencijalno opasni i zahtijevaju liječenje. Posljednjih godina postoji velik interes za korištenjem algoritama dubokog učenja za pomoć u dijagnozi takvih lezija [10].

International Skin Imaging Collaboration (ISIC) nudi opsežan javni skup podataka koji je na raspolaganju akademskoj zajednici za poboljšanje i razvoj klasifikacijskih algoritama, kontinuirano se unaprjeđuje novim podacima [12]. Iako istraživanja koja uključuju umjetnu inteligenciju daju ohrabrujuć napredak u dijagnozi kožnih oboljenja, u skorijoj budućnosti ipak neće zamijeniti medicinske stručnjake. Prije svega, čovjek je potreban za odabir odgovarajuće slike lezije za procjenu, često među stotinama nevažnih. Unatoč tome, algoritmi dubokog učenja nastavit će se razvijati s poboljšanom preciznošću u otkrivanju potencijalno opasnih kožnih oboljenja [10].

# 3. Duboke neuronske mreže

## 3.1. Duboko učenje i neuronske mreže

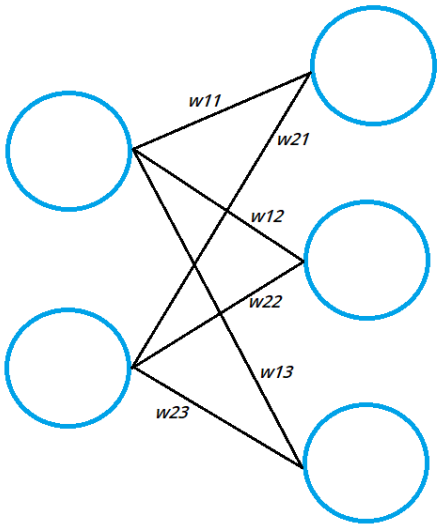
Kao što je već spomenuto, duboko učenje vrsta je strojnog učenja, a nadahnuta je građom ljudskog živčanog sustava. Algoritmi dubokog učenja nastoje donositi zaključke nalik ljudskima kontinuiranom analizom dostupnih podataka koristeći višeslojnu logičku strukturu, duboku neuronsku mrežu. Baš kao što mozak koristimo za identifikaciju obrazaca i klasifikaciju velike količine informacija, neuronske mreže možemo "naučiti" da obavljaju istu funkciju nad nama interesantnim podatkovnim skupovima. Pojedinačne slojeve neuronskih mreža možemo shvatiti kao vrstu filtra koji djeluje od grubog do sve sitnijeg, kako bi postupno povećao vjerojatnost otkrivanja i predviđanja točnih rezultata. Neuronske mreže mogu se koristiti za izvršavanje mnogih zadataka, poput grupiranja i klasifikacije. Grupiranje se zasniva na razvrstavanju neoznačenih podataka prema sličnosti u prepoznatim podatkovnim uzorcima. Klasifikacijske mreže uče na označenim (engl. *labeled*) podatcima kako bi otkrile veze između podataka i njihovih oznaka, a potom se koriste za klasifikaciju podataka u različite kategorije [2].

### Neuroni

Neuronska mreža sastoji se od zbirke povezanih računskih jedinica, koje nazivamo neuronima. Oni modeliraju biološke neurone u ljudskom mozgu. Neuron je osnovna jedinica neuronskih mreža koja prima podatke, nad njima izvodi jednostavne izračune i rezultate koje dobije prosljeđuje dalje. Razlikujemo tri vrste neurona:

- Ulazni neuroni – primaju podatke iz vanjskog svijeta
- Skriveni neuroni – obrađuju te informacije
- Izlazni neuroni – donose krajnji zaključak [2]

Svaka poveznica između dva neurona ima svoju težinu (engl. *weight*). Težine služe za određivanje značajnosti veze među neuronima. Posljedično će rezultati neurona s većom težinom biti dominantniji za izračune u sljedećem neuronu, dok će podatci iz neurona s manjim težinama imati manju važnost, ili se uopće neće prenositi [2].




**Slika 3.1:** Veze između neurona

Na slici 3.1 svaka veza između dva neurona ima određenu numeričku vrijednost i označena je s $w$. Prva vrijednost indeksa označava broj neurona u sloju iz kojeg veza potječe, dok druga vrijednost označava broj neurona u sloju do kojeg veza dolazi. Težine između dva sloja mogu se prikazati matricom težina [2].

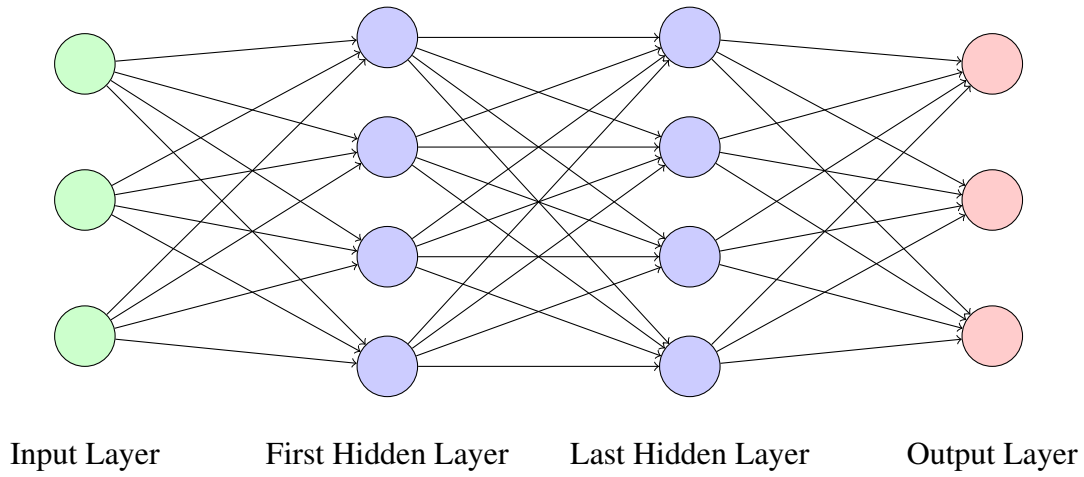
$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \tag{3.1}$$

Matrica težina (3.1) sadrži numeričke vrijednosti za veze između dva sloja neurona u mreži. Dimenzije te matrice odgovaraju broju neurona u dva povezana sloja. Broj redaka odgovara broju neurona u sloju iz kojeg veze potječu, dok broj stupaca odgovara broju neurona u sloju do kojeg te veze vode [2].

## Arhitektura neuronske mreže

Neuroni su grupirani u računske jedinice koje nazivamo slojevima. Slojevi su međusobno povezani i zajedno čine jedinstvenu strukturu mreže. Neuronske mreže obično se sastoje od 3 tipa slojeva:

- Ulazni sloj – prima podatke iz kojih neuronska mreža uči
- Skriveni slojevi – nalaze se između ulaznog i izlaznog sloja, ovdje se obavljaju svi izračuni u mreži
- Izlazni sloj – stvara izlaz za zadane ulazne podatke [3]




**Slika 3.2:** Arhitektura neuronske mreže

Uobičajena arhitektura neuronske mreže i raspored prethodno navedenih slojeva prikazani su na slici 3.2. Ulazni sloj prima ulazne podatke u obliku vektora značajki $\mathbf{x}$. Ako za primjer uzmemo klasifikaciju slika kože, značajke vektora $\mathbf{x}$ bi predstavljale pojedine piksele u predanim slikama. U ulaznom sloju broj neurona odgovara broju značajki u vektoru $\mathbf{x}$. Izlazni sloj daje vektor predviđanja $\mathbf{y}$ koji predstavlja rezultat do kojeg je neuronska mreža došla. Elementi tog vektora predstavljaju vrijednosti neurona u izlaznom sloju mreže. Za navedeni primjer klasifikacije, svaki neuron u izlaznom sloju predstavljao bi drugačiju klasu, tj. prepoznato kožno oboljenje. Kako bi se taj vektor predviđanja dobio, u skrivenim slojevima potrebno je obaviti određene matematičke operacije [2].

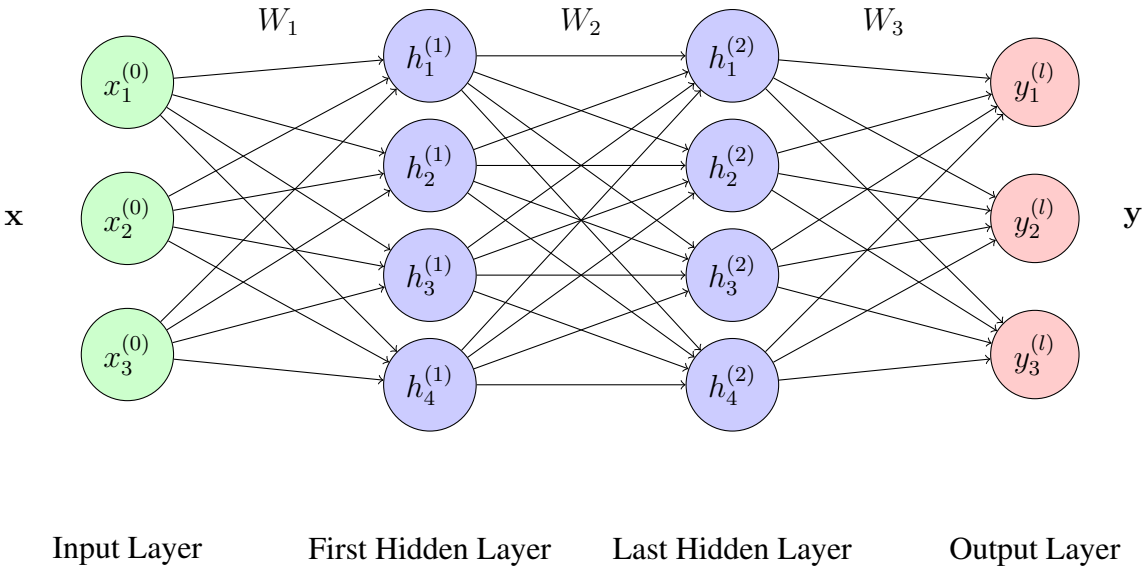
## Učenje neuronske mreže

U procesu učenja neuronske mreže, težine između neurona se mijenjaju. Možemo reći da se cjelokupni proces učenja temelji na pronalaženju optimalnih vrijednosti težina među neuronima. Dakako, skup težina koji ćemo htjeti postići razlikovat će se ovisno o početnim ulaznim podatcima i zadatku koji želimo da je mreža u stanju obavljati. U nastavku je opisan proces učenja neuronske mreže [2].

Za zadani ulazni vektor značajki $\mathbf{x}$, neuronska mreža računa vektor predviđanja $\mathbf{h}$. Taj korak se naziva i "širenje naprijed" (engl. *forward propagation*). Obavljamo produkt ulaznog vektora $\mathbf{x}$ i matrice težina između dva sloja $\mathbf{W}$ (3.2) [2].

$$\begin{aligned}
\mathbf{x}^T\mathbf{W} &= \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \\
&= \begin{pmatrix} x_1 w_{11} + x_2 w_{21} & x_1 w_{12} + x_2 w_{22} & x_1 w_{13} + x_2 w_{23} \end{pmatrix} \\
&= \begin{pmatrix} z_1 & z_2 & z_3 \end{pmatrix} \\
&= \mathbf{z} \\
\mathbf{h} &= \sigma(\mathbf{z})
\end{aligned} \tag{3.2}$$

Rezultat tog produkta je vektor $\mathbf{z}$, a konačni vektor predviđanja $\mathbf{h}$ dobiva se primjenom tzv. aktivacijske funkcije (engl. *activation function*) na vektor $\mathbf{z}$. Aktivacijska funkcija u jednadžbi 3.2 označena je simbolom $\sigma$. U području dubokog učenja najčešće korištene aktivacijske funkcije su $tanh$, $sigmoid$ i $ReLu$. U procesu učenja vrijednosti elemenata vektora $\mathbf{z}$, $\mathbf{h}$ i $\mathbf{y}$ mijenjaju se s promjenama težina u mreži. Ovo opažanje može se proširiti na model neuronske mreže s više slojeva, gdje će se vrijednosti elemenata spomenutih vektora računati za svaki unutarnji sloj mreže [2]. Primjer takve neuronske mreže prikazan je na slici 3.3.




**Slika 3.3:** Prikaz vektora u mreži s više slojeva

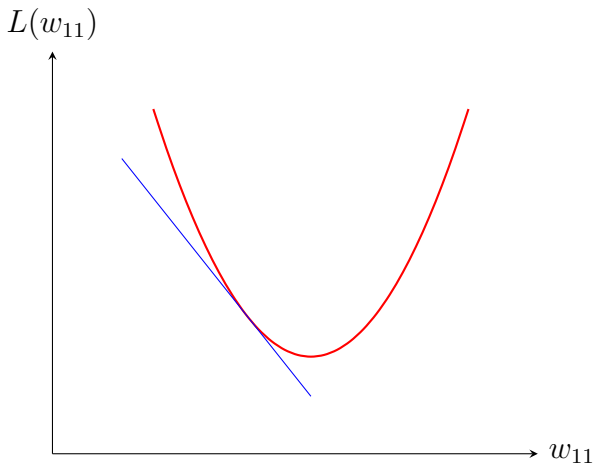
## Funkcija gubitka i gradijentni spust

Nakon računanja vektora predviđanja neuronske mreže $\mathbf{y}$, on se uspoređuje s vektorom $\hat{\mathbf{y}}$ koji sadrži stvarne vrijednosti kojima su podaci označeni. Vrijednost funkcije gubitka (engl. *loss function*) ovisi o razlici ova dva vektora. Ta funkcija zapravo daje numeričku ocjenu o točnosti načina na koji model klasificira. Tako minimizacija vrijednosti funkcije gubitka vodi do točnijih predviđanja neuronske mreže. Za dobre rezultate u procesu učenja mreže potrebno je izabrati odgovarajuću funkciju gubitka. Primjer široko prihvaćene funkcije gubitka je funkcija križne entropije (engl. *Cross-Entropy Loss function*) (3.3) [2].

$$L(\theta) = -\sum_{i=0}^{N} \hat{y}_i \cdot \log(y_i) \tag{3.3}$$

Cilj cijelog procesa učenja neuronske mreže je pronalazak skupa težina za koje je vrijednost funkcije gubitka što je moguće manja. Metoda minimiziranja funkcije gubitka postiže se matematičkom metodom gradijentnog spusta [2].

Tijekom gradijentnog spusta, koristi se gradijent funkcije gubitka za korigiranje težina u neuronskoj mreži.
Na slici 3.4 $x$-koordinata prikazuje vrijednost težine $w_{11}$, dok $y$-koordinata prikazuje vrijednosti funkcije gubitka koja ovisi o toj težini. Računanjem gradijenta te funkcije dobiva se prikazana tangenta koja pokazuje u smjeru najvećeg porasta funkcije gubitka i odgovarajućeg parametra na $x$-osi. Ako se sada taj vektor pomnoži s -1, dobiva se




**Slika 3.4:** Metoda gradijentnog spusta

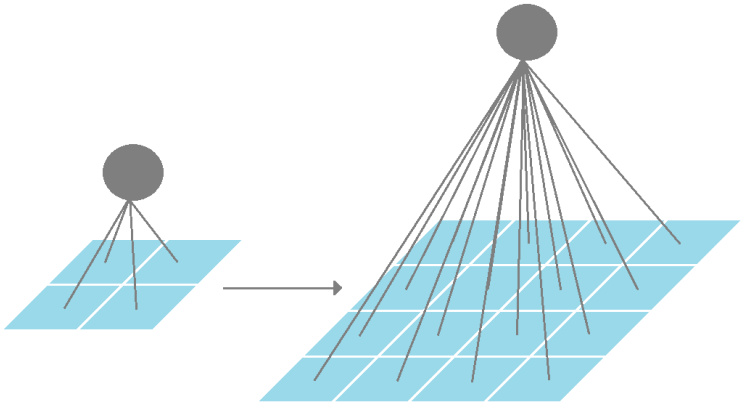
smjer najbržeg pada funkcije gubitka, i upravo u tom smjeru se pomiče vrijednost težine $w_{11}$. Ako se tako vrijednosti svih težina u mreži postupno dovedu do minimuma njihovih funkcija gubitka, u konačnici se dobivaju točnije predikcije za ulazni skup podataka [2].

## 3.2. Konvolucijske neuronske mreže

Područje računalnog vida jedno je od najatraktivnijih područja u umjetnoj inteligenciji. Za cilj ima omogućiti strojevima da svijet vide kao i ljudi, percipiraju ga na sličan način i koriste stečena znanja kako bi mogli prepoznavati, analizirati i klasificirati slike. Napredak u tom području oslanja se na algoritme dubokog učenja, prvenstveno na algoritam konvolucijskih neuronskih mreža [4]. Upravo konvolucijska neuronska mreža bit će korištena za potrebe klasifikacije spomenutih kožnih oboljenja iz slika kože, no prvo će biti opisana općenita ideja takve mrežne arhitekture.

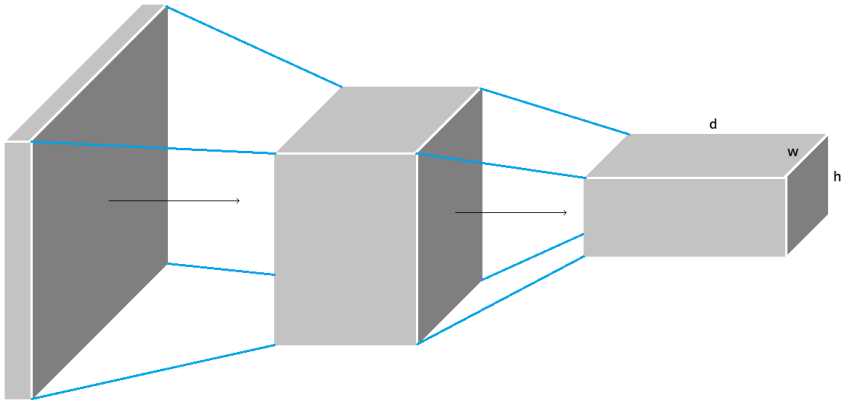
Glavna motivacija za korištenje dubokih neuronskih mreža u području računalnog vida je zaobilaženje dugog postupka izdvajanja značajki. Duboke neuronske mreže sasvim su prikladne za takav zadatak, uzimajući u obzir da kroz slojeve mogu postupno izdvajati značajke ulaznog skupa podataka. Prva pomisao mogla bi biti korištenje potpuno povezanih slojeva (engl. *fully connected layer*) čiji bi neuroni bili povezani sa svakim od ulaznih piksela, što znači da bi u slučaju crno bijele slike dimenzija 20 x 20 piksela značilo da je jedan neuron povezan s 400 težina, a u slučaju slike u boji s tri panela u boji (crvena, zelena i plava) veličine 200 x 200 piksela s 200 x 200 x 3 = 120 000 težina. To ubrzo dovodi do prevelike složenosti što dovodi do prenaučenosti

mreže (engl. *overfitting*), tj. mreža će pokazivati dobre rezultate na skupu podataka za učenje, no neće znati klasificirati testne primjere sa zadovoljavajućom točnošću [4]. Ilustracija naglog povećanja složenosti povećanjem dimenzija ulazne slike prikazana je na slici 3.5.



**Slika 3.5:** Broj veza između slojeva raste povećanjem dimenzija slika

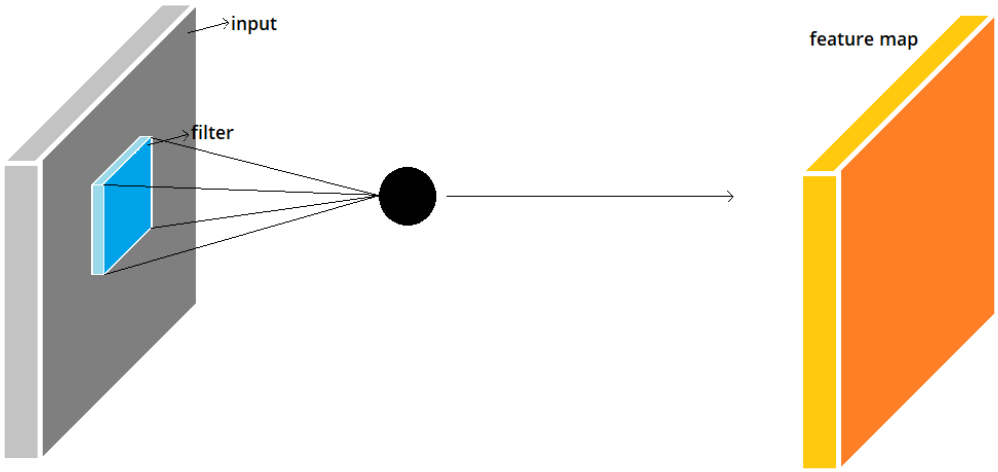
Inspirirani načinom na koji ljudsko oko percipira prostor, slojevi konvolucijske neuronske mreže podijeljeni su u tri dimenzije pa imaju širinu, visinu i dubinu, stoga mogu obrađivati trodimenzionalne ulazne podatke prikazane na slici 3.6. Neuroni u tim slojevima povezani su samo s manjim dijelom neurona iz prethodnog sloja. Tako konvolucijske neuronske mreže drastično smanjuju broj parametara u mreži te pretvaraju sliku u oblik koji je lakše obraditi, bez gubitka značajki važnih za ispravnu klasifikaciju [4]. To je vrlo važno ako želimo izbjeći problem prenaučenosti, u konačnici dobivamo mrežu koja je stabilna ne samo u klasifikaciji podataka skupa za učenje, nego i novih ulaznih skupova podataka.



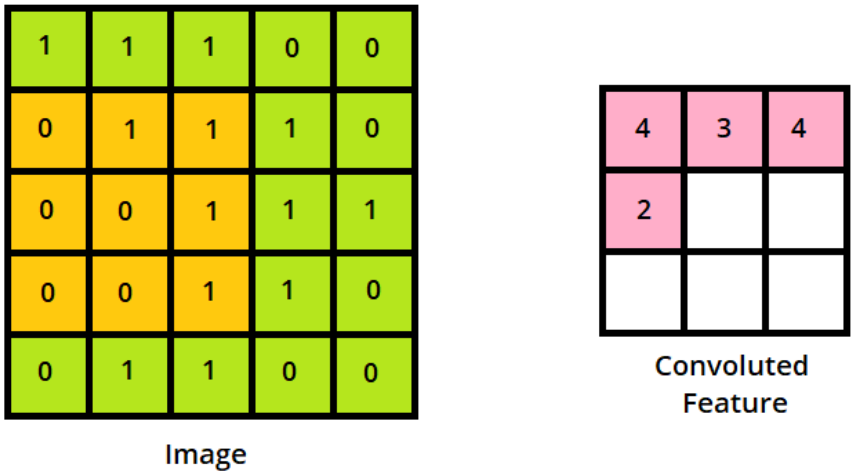

**Slika 3.6:** Tri dimenzije podataka u slojevima konvolucijske neuronske mreže

## Konvolucijski sloj

Bitan koncept za razvoj konvolucijske mrežne arhitekture je filtar, koji je u osnovi detektor značajki na slici. Postupak konvolucije zasniva se na prolasku filtra određenih dimenzija po cijeloj ulaznoj slici. Tako različiti filtri izdvajaju različite specifične značajke u tzv. mape značajki (engl. *feature maps*) [4]. Na slici 3.7 plavom bojom prikazan je filtar koji iz ulazne slike, prikazane sivom, izdvaja značajke i sprema ih u mapu značajki, prikazanu narančastom. Primjenom više filtara, konvolucijska neuronska mreža stvara više mapa značajki i u stanju je uspješno detektirati prostorne i vremenske ovisnosti na slici [5].



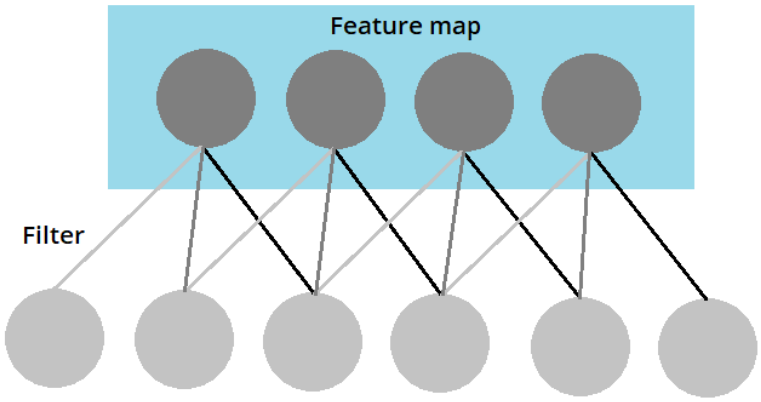

**Slika 3.7:** Izdvajanje značajki u mapu




**Slika 3.8:** Postupak konvolucije

Na slici 3.8 zelenim poljima prikazana je ulazna slika (I) dimenzija 5x5, dok je žutim poljima prikazan konvolucijski filtar (K) dimenzija 3x3. Rozim poljima prikazana je do tad konstruirana mapa značajki. Kako bi prošao po cijeloj ulaznoj slici filtar se pomiče 9 puta, pri čemu svaki put obavlja operaciju matričnog množenja između K i dijela slike P nad kojim se u tom trenutku nalazi. Općenito, filtar se pomiče po ulaznoj slici s određenom duljinom koraka (engl. *stride length*) s lijeva na desno dok ne dođe do ruba slike, kada se pomiče na početak sljedećeg retka. Postupak se ponavlja sve dok se ne prođe cijela površina ulazne slike [5].

U smislu građe konvolucijske neuronske mreže, slojevi neurona predstavljaju izvornu sliku ili mape izdvojenih značajki, dok filtri predstavljaju kombinacije određenih težina među slojevima neurona. Filtar i mapa značajki u tom kontekstu prikazani su na slici 3.9. Neuron u mapi značajki se aktivira ako filtar zadužen za njegovu aktivnost detektira pojavu tražene značajke u prethodnom sloju neurona [4].

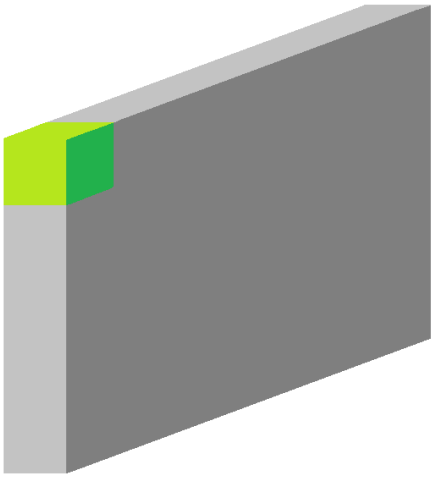



**Slika 3.9:** Mapa značajki i filtar u konvolucijskoj neuronskoj mreži

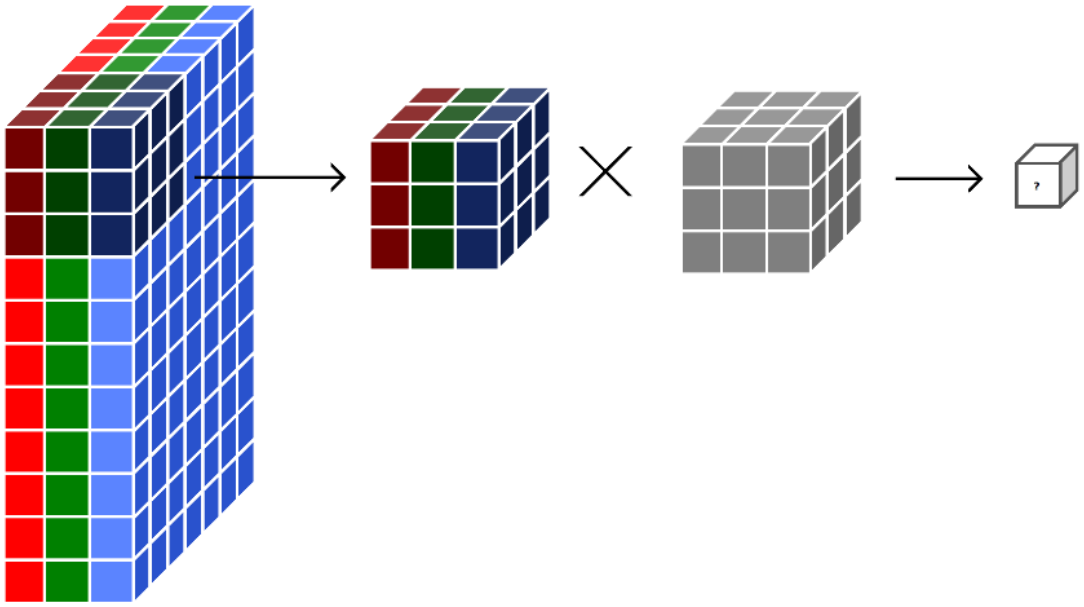
Filtri u konvolucijskim neuronskim mrežama mogu aktivirati značajke iz svih mapa značajki u određenom sloju, što je od posebne važnosti kod ulaznih slika u boji. Naime, kod takvih slika svaki piksel je prikazan kao kombinacija crvene, zelene i plave (engl. *RGB-red, green, blue*) pa će se svaka ulazna slika zapravo dijeliti u tri, gdje svaki pojedini dio predstavlja jednu boju. Stoga su mape značajki u 3D obliku, a stvaraju ih 3D filtri koji u ovom slučaju prolaze cijelim volumenom ulazne slike kako bi ih generirali. Ilustracija takvog filtra u 3D ulaznoj slici prikazana je na slici 3.10, a njegova primjena na trodimenzionalnu RGB sliku prikazana je na slici 3.11. Konvolucijski sloj sastoji se od skupine filtara pomoću kojih ulazni volumen pretvara u izlazni. Dubina filtra odgovara dubini ulazne slike, tako da filtar može kombinirati informacije

o svim naučenim značajkama. Dubina izlaznog volumena konvolucijskog sloja odgovara broju filtara u tom sloju, jer svaki filtar stvara svoj vlastiti izlazni volumen [4].

Zadaća konvolucijskog sloja je izdvajanje značajki visoke razine iz ulazne slike. Konvolucijske neuronske mreže obično imaju više ovakvih slojeva. Prvi konvolucijski sloj iz ulazne slike izdvaja značajke niske razine kao što su rubovi, boja i orijentacija gradijenta. Daljnji slojevi sve se više prilagođavaju značajkama više razine, stvarajući u konačnici mrežu koja je u stanju analizirati i klasificirati slike s visokom točnošću [5].



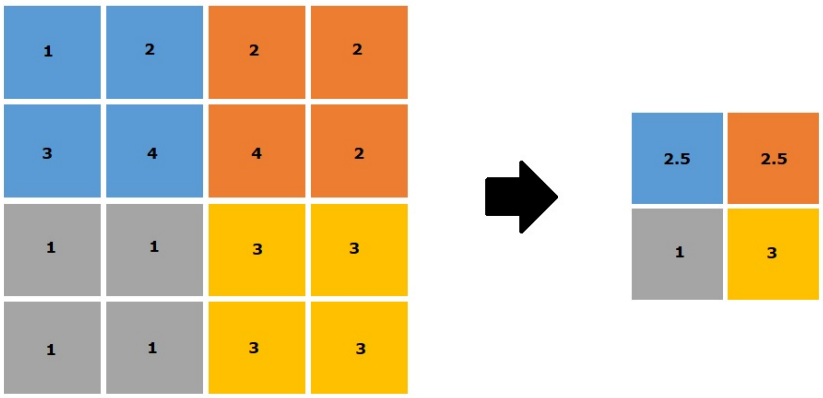
**Slika 3.10:** Skica 3D filtra u trodimenzionalnoj ulaznoj slici



**Slika 3.11:** Primjena 3D filtra na ulaznu RGB sliku u 3D formatu

## Sloj sažimanja

Sloj sažimanja (engl. *pooling layer*) uobičajeno je koristiti nakon konvolucijskih slojeva kako bi se smanjila prostorna složenost stvorenih mapa značajki. Smanjenjem dimenzionalnosti podataka u mreži smanjuje se vrijeme i količina računalnih resursa potrebna za njihovu obradu, što je važno u procesu učenja mreže. Glavna ideja iza sloja sažimanja je razlomiti mapu značajki na jednake cjeline i potom za svaku od cjelina generirati jedno polje u sažetoj mapi značajki, što značajno utječe na smanjenje broja parametara u mreži. Ilustracija primjene sloja sažimanja na generiranoj mapi značajki prikazana je na slici 3.12. Ovakav postupak iz mapa značajki dodatno izdvaja one najdominantnije, čime u konačnici stvaramo veće šanse za izgradnju robusnijeg modela koji će biti manje sklon problemu prenaučenosti [5].

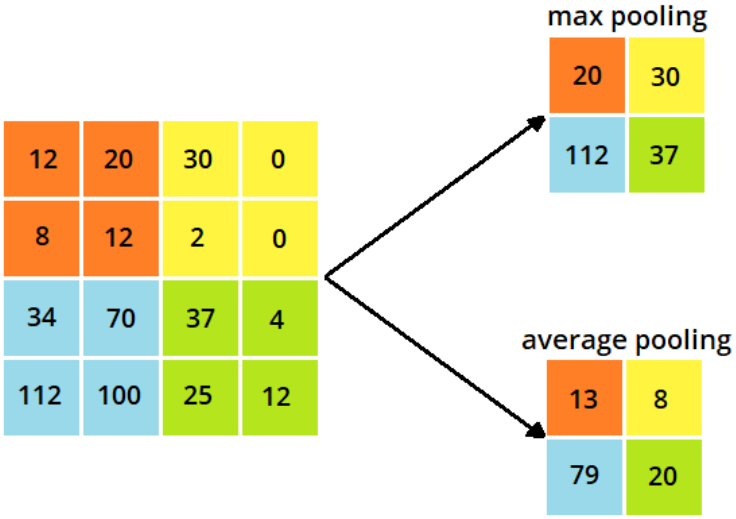



**Slika 3.12:** Primjena sloja sažimanja na generiranoj mapu značajki

Postoje dvije vrste sloja sažimanja:

- Sažimanje maksimalnom vrijednosti (engl. *max pooling*) – iz cjeline u mapi značajki izdvaja se maksimalna vrijednost
- Sažimanje srednjom vrijednosti (engl. *average pooling*) – iz cjeline u mapi značajki izdvaja se srednja vrijednost [5]

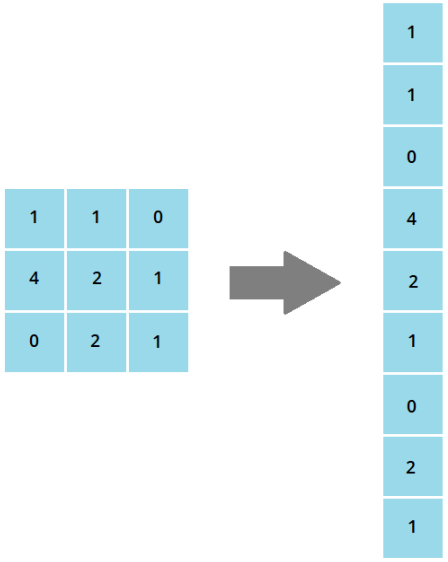
Sažimanje maksimalnom vrijednošću uz smanjenje dimenzionalnosti u potpunosti uklanja bučne aktivacije u mapi značajki, dok sažimanje srednjom vrijednosti obavlja samo smanjenje dimenzionalnosti. Stoga sažimanje maksimalnom vrijednošću općenito daje bolje rezultate u procesu učenja [5]. Opisane vrste slojeva sažimanja prikazane su na slici 3.13.




**Slika 3.13:** Sažimanje maksimalnom i srednjom vrijednošću

## Potpuno povezani sloj

Potpuno povezani slojevi (engl. *fully connected layers*) čine nekoliko zadnjih slojeva konvolucijske neuronske mreže. Ulaz u neurone potpuno povezanog sloja je izlaz iz zadnjeg konvolucijskog ili sloja sažimanja, na koji je dodan sloj neurona za izravnavanje (engl. *flatten layer*). Sloj za izravnavanje izlaz iz konvolucijskog ili sloja sažimanja u obliku trodimenzionalne matrice pretvara u jednodimenzionalni vektor [5]. Takva pretvorba prikazana je na slici 3.14.




**Slika 3.14:** Primjena sloja za izravnavanje prije potpuno povezanog sloja

Ovakav jednodimenzionalni vektor dalje je povezan s nekoliko potpuno povezanih slojeva. U njima se obavljaju svi izračuni karakteristični za potpuno povezane neuronske mreže, uz korištenje $ReLu$ aktivacijske funkcije. Posljednji potpuno povezani sloj potom koristi $Softmax$ aktivacijsku funkciju pomoću koje se na izlazu iz mreže dobivaju vjerojatnosti pripadnosti ulazne slike jednoj od klasa. Na taj se način konvolucijske neuronske mreže koriste za potrebe klasifikacije [13]. Uobičajena arhitektura konvolucijske neuronske mreže sastoji se od nekoliko uzastopnih slojeva konvolucije i sažimanja, nakon kojih dolazi sloj za izravnavanje i nekoliko potpuno povezanih slojeva.

# 4. Izgradnja modela duboke neuronske mreže

Kao što je već spomenuto u uvodnom dijelu, cilj je izgradnja modela duboke neuronske mreže koja će moći klasificirati kožna oboljenja temeljem slika kože. U idućim potpoglavljima bit će opisan postupak razvoja i implementacije takvog modela.

## 4.1. Korištene tehnologije i alati

Za potrebe implementacije modela korišten je programski jezik Python i vanjske biblioteka Keras, NumPy i Pandas. Keras je Python biblioteka otvorenog koda (engl. *open source*) za razvoj i evaluaciju modela dubokog učenja. Služi kao sučelje za numeričku računsku Python biblioteku TensorFlow i omogućava pojednostavljenu definiciju i učenje modela dubokih neuronskih mreža [14]. NumPy je Python biblioteka koja pojednostavljuje rad s n-dimenzionalnim poljima što je od velike važnosti pri manipulaciji velikim količinama podataka korištenim za modele dubokog učenja [15]. Pandas je Python biblioteka otvorenog koda korištena za analizu i učitavanje podataka, a razvijena je kao dio NumPy biblioteke [16].

Za učenje i razvoj modela duboke neuronske mreže korištena je platforma Google Colab koja osim pisanja i izvođenja Python koda izravno u pregledniku nudi i besplatan pristup dodatnoj sklopovskoj potpori za potrebe učenja neuronskih mreža.

## 4.2. Podaci

Prvi korak je prikupljanje prikladnih podataka, tj. slika kože pomoću kojih će model učiti. Za ovaj zadatak odabran je $HAM10000$ (engl. *"Human Against Machine with* $10000$ *training images"*) skup podataka koji se sastoji od $10015$ slika kožnih lezija. Taj skup podataka je javan, a objavio ga je "Harvard database" u lipnju 2018. godine. Više od 50% lezija potvrđene su histopatologijom (histo), istinita oznaka za ostala oboljenja (engl. *ground truth*) temelji se na naknadnim dermatološkim pregledima (follow_up), konsenzusu stručnjaka (konsenzus) ili potvrdi in-vivo konfokalnom mikroskopijom (confocal) [12].

U dijelu 2.2 opisane su sve klase kožnih oboljenja koja su prisutna u skupu podataka. Kratice klasa tih oboljenja korištene u datoteci s metapodacima su sljedeće:

1. Melanocytic nevi – nv
2. Melanoma – mel
3. Benign keratosis-like lesions – bkl
4. Basal cell carcinoma – bcc
5. Actinic keratoses – akiec
6. Vascular lesions – vas
7. Dermatofibroma – df

### Metapodaci

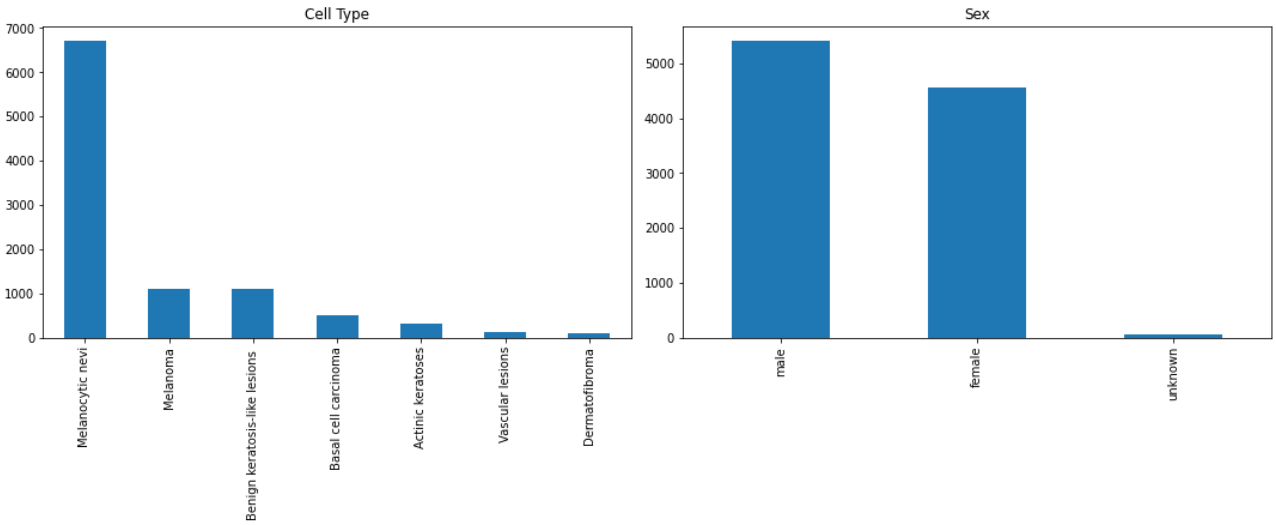
Uz slike kožnih oboljenja, u skupu podataka $HAM10000$ priložena je i datoteka s metapodacima $HAM10000\_metadata.csv$. U toj datoteci priložene su dodatne informacije o skupu podataka. Najvažniji podatak za potrebe izgradnje klasifikacijskog modela svakako je oznaka klase kožnog oboljenja za svaku sliku. Ostali metapodaci neće biti od tolike važnosti za samu klasifikaciju, no svakako ih valja proučiti kako bi se stekao bolji dojam o prirodi podataka koji će se koristiti za izgradnju modela. Na slici 4.1 dan je uvid u strukturu metapodataka dostupnih u navedenoj datoteci.

| | lesion_id | image_id | dx | dx_type | age | sex | localization |
|---|---|---|---|---|---|---|---|
| **0** | HAM_0000118 | ISIC_0027419 | bkl | histo | 80.0 | male | scalp |
| **1** | HAM_0000118 | ISIC_0025030 | bkl | histo | 80.0 | male | scalp |
| **2** | HAM_0002730 | ISIC_0026769 | bkl | histo | 80.0 | male | scalp |
| **3** | HAM_0002730 | ISIC_0025661 | bkl | histo | 80.0 | male | scalp |
| **4** | HAM_0001466 | ISIC_0031633 | bkl | histo | 75.0 | male | ear |

**Slika 4.1:** Metapodaci iz datoteke $HAM10000\_metadata.csv$

U metapodacima su značajni stupci koji označavaju kraticu klase kožne lezije (dx), tip istinite oznake za oboljenje (dx_type), godine (age) i spol (sex) bolesnika te lokalizacija lezije na tijelu (localization). Stupac za id slike (image_id) bit će korišten za stvaranje putanje do slike oboljenja u učitanim podacima.

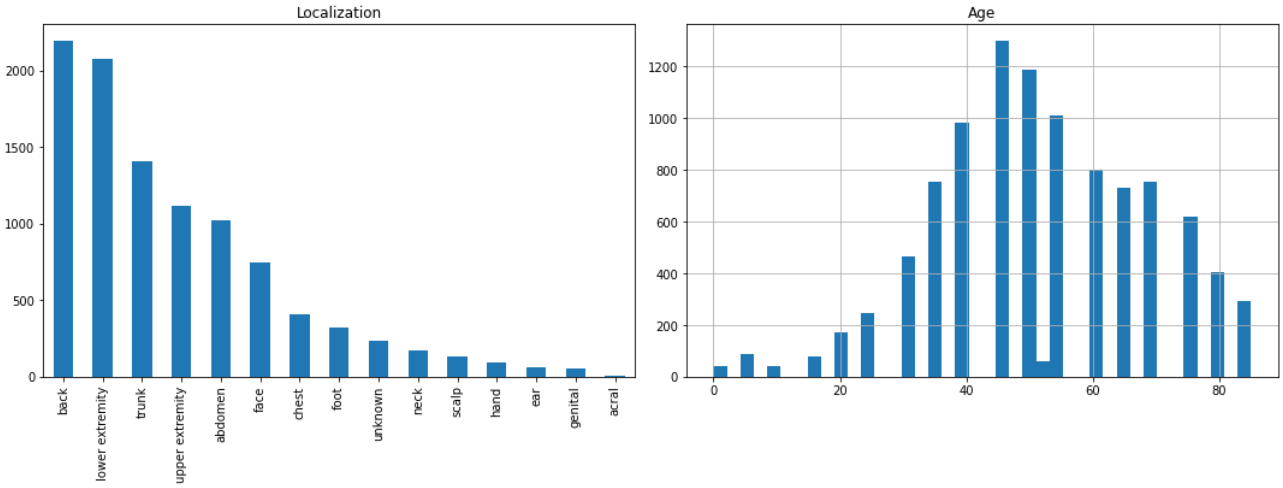
Korištenjem ovih metapodataka moguće je podrobnije istražiti osobine promatranog skupa podataka. Kao što je već spomenuto, u konkretnom klasifikacijskom modelu koristit će se samo podatak o klasi kožne lezije, no ostale metapodatke moguće je iskoristiti za daljnje studije vezane za proučavanje populacije. U nastavku su prikazane raspodjele podataka po dostupnim metapodacima.




**Slika 4.2:** Distribucija klasa kožnih lezija i spola pacijenata u skupu podataka

Iz distribucije po klasama kožnih lezija prikazane na slici 4.2, vidljivo je da postoji velik disbalans u broju slika po pojedinoj klasi. Prisutno je 6705 slika oboljenja klase "Melanocytic nevi", dok slika oboljenja klase "Dermatofibroma" ima svega 115. Ovakve pojave su vrlo česte u skupovima podataka za medicinska istraživanja zbog

ograničenog broja pacijenata i učestalosti pojavljivanja određene bolesti u promatranoj populaciji.




**Slika 4.3:** Distribucija lokalizacija kožinih lezija i starosti pacijenata u skupu podataka

Na slici 4.3 prikazane su distribucije po fizičkoj lokalizaciji kožnih lezija, te po starosti promatranih pacijenata. Iz tih distribucija vidljivo je da se kožne lezije najčešće javljaju na leđima i nogama, u srednjoj životnoj dobi.

## 4.3. Priprema podataka

Nakon preuzimanja i raspakiravanja zip arhive s bazom podataka o kožnim oboljenjima, potrebno je obaviti učitavanje i predobradu podataka kako bi se mogli koristiti u gradnji klasifikacijskog modela.

### Učitavanje slika u Dataframe

Nakon učitavanja datoteke s metapodacima ($HAM10000\_metadata.csv$) bibliotekom Pandas, kao povratnu vrijednost dobivamo Pandas Dataframe. Dataframe omogućava lakše baratanje dvodimenzionalnim tabličnim podacima s označenim redovima i stupcima, a dodatno je omogućena promjena i dodavanje podataka u Dataframe [16]. Učitavanje datoteke postižemo navedenom naredbom.

```
skin_dataframe = pd.read_csv(os.path.join(base_skin_dir,
↪  'HAM10000_metadata.csv'))
```

U učitani dataframe potrebno je dodati dva stupca. Stupac $cell\_type$ u kojemu će za svaku sliku biti oznaka punog imena kožne lezije generira se pomoću kratice lezije ($dx$) i Pythonovog rječnika (engl. *dictionary*) $lesion\_type\_dict$. Stupac $cell\_type\_index$ će za svaku sliku pohranjivati indeks pripadajuće oznake kožne lezije u Pythonovu rječniku $lesion\_type\_dict$, što će nam pomoći pri baratanju podacima u kasnijim koracima. Rječnik $lesion\_type\_dict$ i naredba za dodavanje navedenih stupaca u Dataframe dani su u nastavku.

```
lesion_type_dict = {
    'nv': 'Melanocytic nevi',
    'mel': 'Melanoma',
    'bkl': 'Benign keratosis-like lesions ',
    'bcc': 'Basal cell carcinoma',
    'akiec': 'Actinic keratoses',
    'vasc': 'Vascular lesions',
    'df': 'Dermatofibroma'
}

skin_dataframe['cell_type'] =
 ↪  skin_dataframe['dx'].map(lesion_type_dict.get)
skin_dataframe['cell_type_index'] =
 ↪  pd.Categorical(skin_dataframe['cell_type']).codes
```

Idući korak je dodavanje stupca sa slikama ($image$) u Dataframe, no prije toga potrebno je dodati još jedan stupac ($path$) u kojem će za svaku sliku biti zapisana pripadajuća putanja na disku. Putanja pojedine slike dobiva se iz prethodno pripremljenog Pythonovog rječnika $imageid\_path\_dict$, gdje su parovi ključ:vrijednost oblika image_id:putanja, kako je prikazano na slici 4.4.

```
ISIC_0033360, ../content/HAM10000_images_part_2/ISIC_0033360.jpg
ISIC_0030061, ../content/HAM10000_images_part_2/ISIC_0030061.jpg
ISIC_0032359, ../content/HAM10000_images_part_2/ISIC_0032359.jpg
ISIC_0032465, ../content/HAM10000_images_part_2/ISIC_0032465.jpg
ISIC_0033735, ../content/HAM10000_images_part_2/ISIC_0033735.jpg
ISIC_0032846, ../content/HAM10000_images_part_2/ISIC_0032846.jpg
ISIC_0032437, ../content/HAM10000_images_part_2/ISIC_0032437.jpg
ISIC_0029868, ../content/HAM10000_images_part_2/ISIC_0029868.jpg
ISIC_0033416, ../content/HAM10000_images_part_2/ISIC_0033416.jpg
ISIC_0033060, ../content/HAM10000_images_part_2/ISIC_0033060.jpg
```

**Slika 4.4:** Dio sadržaja rječnika *imageid_path_dict*

Slike su izvorno spremljene u *.jpg* formatu dimenzija 450x600x3, što nije prikladno za ulaz u Kerasov model duboke neuronske mreže. Iz tog razloga slikama se dimenzije mijenjaju na 100x75. Konačno, dodavanje stupca *image* u Dataframe uz učitavanje i promjenu veličine slika prikazano je u nastavku, a krajnji izgled dobivenog Dataframea dan je na slici 4.5.

```
skin_dataframe['image'] =
↪  skin_dataframe['path'].map(lambda
↪  x:np.asarray(Image.open(x).resize((100,75))))
```

| | lesion_id | image_id | dx | dx_type | age | sex | localization | path | cell_type | cell_type_index | image |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | HAM_0000118 | ISIC_0027419 | bkl | histo | 80.0 | male | scalp | ../content/HAM10000_images_part_1 /ISIC_0027419... | Benign keratosis-like lesions | 2 | [[[190, 153, 194], [192, 154, 196], [191, 153,... |
| 1 | HAM_0000118 | ISIC_0025030 | bkl | histo | 80.0 | male | scalp | ../content/HAM10000_images_part_1 /ISIC_0025030... | Benign keratosis-like lesions | 2 | [[[23, 13, 22], [24, 14, 24], [25, 14, 28], [3... |
| 2 | HAM_0002730 | ISIC_0026769 | bkl | histo | 80.0 | male | scalp | ../content/HAM10000_images_part_1 /ISIC_0026769... | Benign keratosis-like lesions | 2 | [[[185, 127, 137], [189, 133, 147], [194, 136,... |
| 3 | HAM_0002730 | ISIC_0025661 | bkl | histo | 80.0 | male | scalp | ../content/HAM10000_images_part_1 /ISIC_0025661... | Benign keratosis-like lesions | 2 | [[[24, 11, 17], [26, 13, 22], [38, 21, 32], [5... |
| 4 | HAM_0001466 | ISIC_0031633 | bkl | histo | 75.0 | male | ear | ../content/HAM10000_images_part_2 /ISIC_0031633... | Benign keratosis-like lesions | 2 | [[[134, 90, 113], [147, 102, 125], [159, 115, ... |

**Slika 4.5:** Dataframe sa dodanim stupcima *cell_type*, *cell_type_index*, *path* i *image*

## Podjela podataka na značajke i ciljne oznake

Još jedan međukorak je izdvajanje ciljnih oznaka učitanih podataka. Stupac *cell_type_index* izdvaja se iz Dataframea i sprema u polje ciljnih oznaka *target*, dok polje značajki *features* čine preostali stupci. To postižemo idućim naredbama.

```
features=skin_dataframe.drop(columns=['cell_type_index'],axis=1)
target=skin_dataframe['cell_type_index']
```

## Transformacija ciljnih oznaka

Polje ciljnih oznaka $target$ potrebno je pretvoriti u zapis $One\ Hot\ Encoding$. Kako modeli dubokog učenja ne mogu raditi sa izvornim kategoričkim podacima, $One\ Hot\ Encoding$ ih pretvara u numerički oblik koji modeli dubokog učenja mogu koristiti. Takav zapis uklanja izbornu oznaku i zamjenjuje ju binarnom ovisno o broju mogućih kategorija. Kako u promatranom skupu podataka postoji 7 klasa kožnih oboljenja, oznake će biti duljine 7 binarnih znakova. Klase kožnih oboljenja sa pripadajućim $One\ Hot\ Encoding$ oznakama prikazane su tablicom 4.1.

```
y_train = to_categorical(y_train_o, num_classes = 7)
y_test = to_categorical(y_test_o, num_classes = 7)
```

**Tablica 4.1:** Tablica One hot encoding oznaka za kožna oboljenja

| Redni broj oznake | Kožna bolest | One hot encoding oznaka |
|---|---|---|
| 1 | Melanocytic nevi | 1 0 0 0 0 0 0 |
| 2 | Melanoma | 0 1 0 0 0 0 0 |
| 3 | Benign keratosis-like lesions | 0 0 1 0 0 0 0 |
| 4 | Basal cell carcinoma | 0 0 0 1 0 0 0 |
| 5 | Actinic keratoses | 0 0 0 0 1 0 0 |
| 6 | Vascular lesions | 0 0 0 0 0 1 0 |
| 7 | Dermatofibroma | 0 0 0 0 0 0 1 |

## Stvaranje skupova za učenje i testiranje

Kako bi se podaci pripremili za implementaciju modela potrebno ih je podijeliti u skupove za učenje, validaciju i testiranje. Model uči na skupu podataka za učenje, određuje optimalne parametre na skupu podataka za validaciju te se testira s određenim optimalnim parametrima na skupu podataka za testiranje. Uobičajeno se podaci u ova tri skupa dijele u omjerima 50%/20%/30%, no podjela inicijalnog skupa podataka napravljena je u omjeru 72%/8%/20%. Takva podjela davala je najbolju točnost ovom modelu, ali i modelima u projektima koji koriste isti skup podataka [21].

```
x_train_o, x_test_o, y_train_o, y_test_o =
 ↪  train_test_split(features, target,
 ↪  test_size=0.20,random_state=555)

x_train, x_validate, y_train, y_validate =
 ↪  train_test_split(x_train, y_train, test_size = 0.1,
 ↪  random_state = 55)
```

## 4.4. Gradnja modela

Model duboke neuronske mreže izgrađen je prema načelima konvolucijske mrežne arhitekture, koja je prethodno teorijski opisana u poglavlju 3.2. Korišten je Keras Sequential API [14] koji omogućava apstrahiranu gradnju mreže sloj po sloj. Na početku se nalazi ulazni (engl. *input*) sloj za učitavanje podataka, dimenzija $75$x$100$x$3$. Slijede tri sloja konvolucije. Prvi se sastoji od jednog konvolucijskog sloja i sloja sažimanja, a druga dva od dva konvolucijska sloja nakon kojih dolazi sloj sažimanja. Kao optimalna veličina filtra u konvolucijskim slojevima pokazala se veličina $3$x$3$, a u slojevima sažimanja korišteno je sažimanje maksimalnom vrijednošću s veličinom površine sažimanja $2$x$2$. Nakon konvolucijskih slojeva primijenjen je sloj za izravnavanje i $3$ potpuno povezana sloja s aktivacijskom funkcijom $ReLu$. Zadnji sloj je izlazni s aktivacijskom funkcijom $Softmax$. Slojevi razvijenog modela prikazani su na slici 4.6. Baza za oblikovanje modela bila je uobičajena arhitektura konvolucijske neuronske mreže s dva sloja konvolucije (konvolucijski sloj + sloj sažimanja), slojem izravnavanja i potpuno povezanim slojevima koja se uobičajeno koristi za probleme prepoznavanja slika. Uz takav, isproban je i model s dodatnim slojem konvolucije kako bi se smanjio broj parametara u mreži. Takav model postizao je veću točnost na skupu za validaciju i testiranje, a imao je i manju razliku u točnosti skupa za učenje i validaciju, tj. bio je manje sklon prenaučenosti.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 75, 100, 16)       448
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 100, 32)       4640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 37, 50, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 50, 32)        9248
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 37, 50, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 18, 25, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 18, 25, 32)        18464
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 18, 25, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 9, 12, 64)         0
_________________________________________________________________
flatten (Flatten)            (None, 6912)              0
_________________________________________________________________
dense (Dense)                (None, 128)               884864
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 231
=================================================================
Total params: 965,223
Trainable params: 965,223
Non-trainable params: 0
_________________________________________________________________
```

**Slika 4.6:** Izgrađeni model

## 4.5. Učenje i testiranje modela

Prije pokretanja procesa učenja, potrebno je definirati optimizacijsku funkciju (engl. *optimizer*), funkciju gubitka (engl. *loss function*) i mjeru kojom ćemo tijekom učenja pratiti performanse modela (engl. *metrics*). Optimizacijska funkcija iterativno mijenja vrijednosti parametara u neuronskoj mreži u cilju smanjivanja vrijednosti funkcije gubitka. Korištena je optimizacijska funkcija $Adam$ jer održava pojedinačne stope učenja po parametru u neuronskoj mreži. Funkcija gubitka mjeri koliko dobro model predviđa označene podatke i mjeri stopu pogreške između stvarnih i predviđenih oznaka. Za probleme klasifikacija s više od dvije klase koristi se funkcija gubitka kategoričke križne entropije (engl. *categorical_crossentropy*). Za mjeru učinkovitosti korištena je preciznost predviđanja modela na validacijskom skupu podataka (engl. *accuracy*).

```
optimizer = Adam(learning_rate=0.001, beta_1=0.9,
 ↪  beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)

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

Bitan pojam je i stopa učenja (engl. *learning rate*) koja je mjera brzine kojom optimizacijska funkcija mijenja parametre modela. Što je stopa učenja veća optimizacijska funkcija za veću stopu mijenja parametre, što može dovesti do upadanja u lokalni minimum funkcije gubitka. Pokazalo se da je korisno postupno smanjivati stopu učenja. To se postiže uporabom funkcije $ReduceLROnPlateau$ iz paketa $Keras.Callbacks$ [14], ta funkcija smanjuje stopu učenja za pola ako se preciznost modela ne povećava nakon tri epohe.

```
lr_reduction = ReduceLROnPlateau(monitor='val_accuracy',
                                 patience=3,
                                 verbose=1,
                                 factor=0.5,
                                 min_lr=0.00001)
```

Konačno, možemo pokrenuti proces učenja, a potom i testiranja modela. Ispis procesa učenja prikazan je na slici 4.7, a procesa testiranja na slici 4.8.

```
Epoch 1/20
721/721 [==============================] - 188s 240ms/step - loss: 1.0632 - accuracy: 0.6725 - val_loss: 0.9026 - val_accuracy: 0.6870
Epoch 2/20
721/721 [==============================] - 170s 236ms/step - loss: 0.8956 - accuracy: 0.6716 - val_loss: 0.9176 - val_accuracy: 0.6920
Epoch 3/20
721/721 [==============================] - 170s 236ms/step - loss: 0.8311 - accuracy: 0.6925 - val_loss: 0.7834 - val_accuracy: 0.7095
Epoch 4/20
721/721 [==============================] - 170s 236ms/step - loss: 0.7839 - accuracy: 0.7173 - val_loss: 0.8186 - val_accuracy: 0.6820
Epoch 5/20
721/721 [==============================] - 171s 237ms/step - loss: 0.7764 - accuracy: 0.7230 - val_loss: 0.7257 - val_accuracy: 0.7369
Epoch 6/20
721/721 [==============================] - 171s 237ms/step - loss: 0.7544 - accuracy: 0.7226 - val_loss: 0.7317 - val_accuracy: 0.7294
Epoch 7/20
721/721 [==============================] - 170s 236ms/step - loss: 0.7255 - accuracy: 0.7380 - val_loss: 0.7222 - val_accuracy: 0.7332
Epoch 8/20
721/721 [==============================] - 170s 236ms/step - loss: 0.6998 - accuracy: 0.7452 - val_loss: 0.6738 - val_accuracy: 0.7668
Epoch 9/20
721/721 [==============================] - 170s 236ms/step - loss: 0.6676 - accuracy: 0.7577 - val_loss: 0.6517 - val_accuracy: 0.7631
Epoch 10/20
721/721 [==============================] - 170s 236ms/step - loss: 0.6512 - accuracy: 0.7644 - val_loss: 0.7804 - val_accuracy: 0.6920
Epoch 11/20
721/721 [==============================] - 170s 236ms/step - loss: 0.6457 - accuracy: 0.7646 - val_loss: 0.6659 - val_accuracy: 0.7793
Epoch 12/20
721/721 [==============================] - 170s 236ms/step - loss: 0.6329 - accuracy: 0.7695 - val_loss: 0.7391 - val_accuracy: 0.7107
Epoch 13/20
721/721 [==============================] - 170s 236ms/step - loss: 0.6160 - accuracy: 0.7705 - val_loss: 0.6900 - val_accuracy: 0.7544
Epoch 14/20
721/721 [==============================] - 169s 235ms/step - loss: 0.6006 - accuracy: 0.7773 - val_loss: 0.7007 - val_accuracy: 0.7618

Epoch 00014: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 15/20
721/721 [==============================] - 169s 234ms/step - loss: 0.5582 - accuracy: 0.7899 - val_loss: 0.6458 - val_accuracy: 0.7731
Epoch 16/20
721/721 [==============================] - 169s 235ms/step - loss: 0.5348 - accuracy: 0.7986 - val_loss: 0.6577 - val_accuracy: 0.7693
Epoch 17/20
721/721 [==============================] - 169s 235ms/step - loss: 0.5314 - accuracy: 0.7991 - val_loss: 0.6620 - val_accuracy: 0.7781

Epoch 00017: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 18/20
721/721 [==============================] - 170s 235ms/step - loss: 0.4772 - accuracy: 0.8191 - val_loss: 0.6446 - val_accuracy: 0.7768
Epoch 19/20
721/721 [==============================] - 170s 235ms/step - loss: 0.4538 - accuracy: 0.8309 - val_loss: 0.6501 - val_accuracy: 0.7743
Epoch 20/20
721/721 [==============================] - 170s 235ms/step - loss: 0.4455 - accuracy: 0.8333 - val_loss: 0.6425 - val_accuracy: 0.7706
```

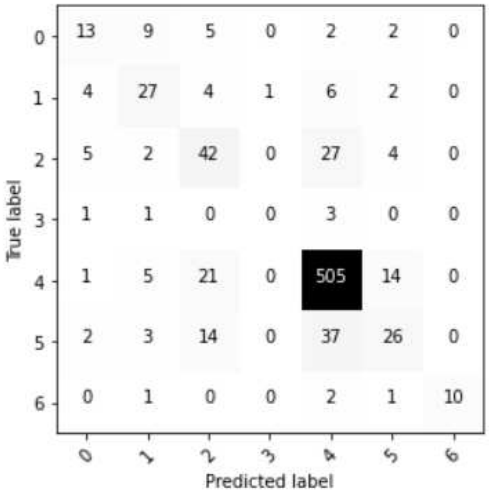
**Slika 4.7:** Proces učenja modela

```
63/63 [==============================] - 9s 139ms/step - loss: 0.6478 - accuracy: 0.7753
26/26 [==============================] - 4s 134ms/step - loss: 0.6425 - accuracy: 0.7706
Validation: accuracy = 0.770574  ;  loss_v = 0.642453
Test: accuracy = 0.775337  ;  loss = 0.647832
```

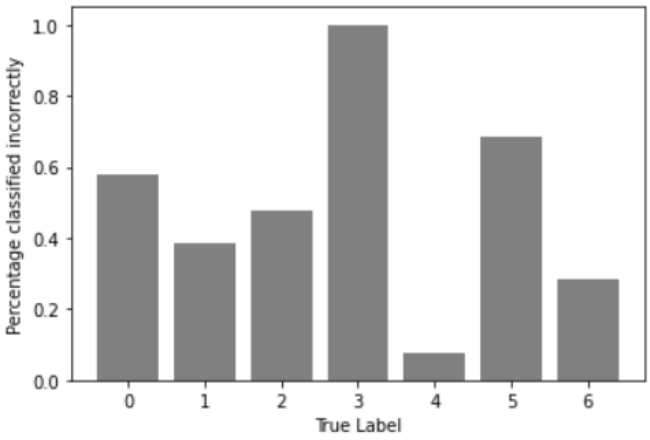
**Slika 4.8:** Proces testiranja modela

Proces učenja sastoji se od 20 epoha, a veličina uzorka (engl. *batch size*) postavljena je na 10. Model postiže konačnu točnost od 88.33% na skupu podataka za učenje, 77.06% na skupu podataka za validaciju i 77.0574% na skupu podataka za testiranje. Kako bi se bolje dočarale performanse razvijenog modela koristi se matrica konfuzije. Matrica kofuzije je tehnika za sumiranje izvedbe klasifikacijskih algoritama po svim klasama podataka. Korisna je jer samo točnost klasifikacije može biti zavaravajuća mjera ako postoji disbalans broja podataka po klasama ili se javljaju više od dvije klase. U skupu podataka $HAM10000$ prisutna su oba slučaja. Na slici 4.2 prikazana je distribucija 7 klasa kožnih lezija među kojima postoji velik disbalans, npr. prisutno je 6705 slika oboljenja klase "Melanocytic nevi", dok slika oboljenja klase "Dermatofibroma" ima svega 115. Posljedično, model je u procesu učenja koristio veći broj

slika najzastupljenijih klasa i prilagođeniji je tim podacima. Tako model radi više grešaka u klasifikaciji podataka koji spadaju u jednu od manje zastupljenih klasa u skupu podataka, što je vidljivo u matrici konfuzije. Matrica konfuzije za razvijeni model na skupu podataka za validaciju prikazana je na slici 4.9.




**Slika 4.9:** Matrica konfuzije




**Slika 4.10:** Udio pogrešnih klasifikacija po klasama kožnih oboljenja

Kako bi matrica konfuzije bila jasnija, na slici 4.10 prikazano je za koje klase model ima najviše pogrešnih klasifikacija na skupu podataka za validaciju. Zabilježeno je najviše pogrešnih klasifikacija za bolesti "Basal cell carcinoma" (oznaka 3) i "Vascular lesions" (oznaka 5), dok je najmanje pogrešnih klasifikacija za bolest "Actinic keratoses" (oznaka 4).

Udio pogrešnih klasifikacija za manje zastupljene klase kožnih oboljenja mogao bi se smanjiti povećavanjem broja slika tih klasa u skupu podataka. To se može pos-

tići manipulacijom postojećim slikama ili dodavanjem novih. Sa stajališta dijagnostike ovakve netočne klasifikacije predstavljaju velik problem, jer bi o njima ovisilo daljnje liječenje i zdravstveno stanje pacijenta. Stoga je potrebno još dugo eksperimentirati i prikupljati podatke kako bi stvarna uporaba ovakvih sustava postala moguća.

## 4.6. Usporedba sa sličnim rješenjima

### ResNet50

Ideja iza rezidualnih mreža (ResNet) je pokušaj prevladavanja problema pri radu s dubljim modelima gdje se pogreška u učenju počinje povećavati s dodavanjem dodatnih slojeva. Ova mrežna arhitektura najčešće je korištena unaprijed naučena mreža za prepoznavanje slika. Model $ResNet50$ koji koristi 50 konvolucijskih slojeva na promatranom skupu podataka postigao je točnosti prikazane slikom 4.11 [6].

```
ResNet50 Training: accuracy = 0.814147
ResNet50 Validation: accuracy = 0.775561
ResNet50 Test: accuracy = 0.768847
```

**Slika 4.11:** Postignute točnosti modela $ResNet50$ [6]

# 5. Izgradnja aplikacije

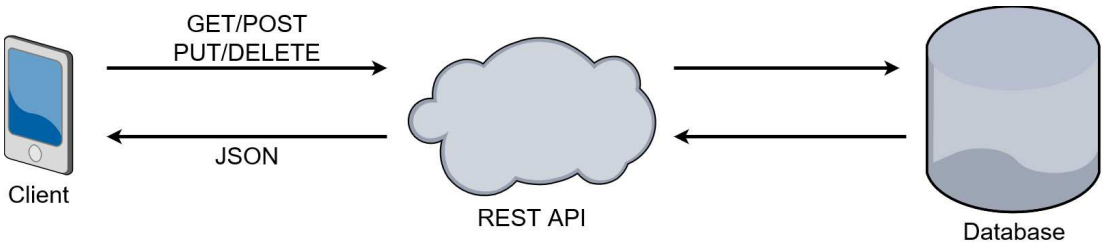
Nakon izgradnje modela duboke neuronske mreže bilo je potrebno izgraditi mobilnu aplikaciju koja će pružiti korisničko sučelje za klasifikaciju kožnih oboljenja na temelju slika kože. U idućim potpoglavljima bit će opisan postupak razvoja i implementacije takve aplikacije.

## 5.1. Opis sustava

Aplikacija omogućava slikanje kože i prijenos slika na poslužiteljsku stranu gdje se obavlja klasifikacija kožnog oboljenja korištenjem razvijenog modela duboke neuronske mreže. Nakon klasifikacije rezultati se s poslanom slikom spremaju u bazu podataka. Kako se rezultati trajno pohranjuju u bazu podataka, korisniku je u aplikaciji omogućen ispis svih obavljenih klasifikacija. Osim uporabe kamere, omogućen je i izbor iz galerije za odabir postojeće slike za klasifikaciju. Kako bi korisnik mogao pristupiti opisanim funkcionalnostima potrebna je njegova registracija i prijava u sustav, pri čemu se podaci o korisniku također trajno pohranjuju u bazu podataka.

## 5.2. Arhitektura sustava

Sustav je oblikovan u arhitekturalnom stilu REST i ima tri glavna sudionika: korisničku stranu (engl. *frontend*), poslužiteljsku stranu (engl. *backend*) i bazu podataka (engl. *database*). Skica arhitekture razvijenog sustava prikazana je na slici 5.1.




**Slika 5.1:** REST arhitektura sustava

REST (engl. *Representational State Transfer*) je arhitekturalni stil za oblikovanje usluga na webu. Kada klijent pošalje zahtjev za resursima prema REST poslužitelju, poslužitelj vraća trenutno stanje traženih resursa u bazi podataka [7].

U razvijenoj aplikaciji, klijentsku stranu čini mobilni uređaj s pokrenutom klijentskom aplikacijom koja komunicira s poslužiteljskom stranom, tj. REST poslužiteljem. Klijent s REST poslužiteljem komunicira protokolom HTTP, koristeći zahtjeve GET i POST. HTTP je komunikacijski protokol koji omogućava dohvaćanje i slanje resursa na webu. Podaci se dohvaćaju GET zahtjevom, a šalju se koristeći POST zahtjev. Podaci koji se šalju su u JSON formatu. JSON (engl. *JavaScript Object Notation*) je standardni format razmjene i pohrane podataka u obliku parova ključ-vrijednost (engl. *key-value pairs*).

Poslužiteljsku stranu čini pokrenuti REST poslužitelj. Poslužitelj prima HTTP zahtjeve klijentske strane i na temelju traženih resursa formira zahtjeve prema bazi podataka. Poslužitelj s bazom podataka komunicira jezikom SQL. SQL (engl. *Structured Query Language*) je standardni jezik za komunikaciju i upravljanje relacijskim bazama podataka. Temeljem povratne vrijednosti zahtjeva upućenih prema bazi podataka, poslužitelj klijentu vraća trenutno stanje traženih resursa.

Bazu podataka čini pokrenuti sustav za upravljanje bazom podataka (engl. *DBMS-Database Management System*) koji omogućava trajnu pohranu i pristup podacima u bazi.

## 5.3. Korištene tehnologije i alati

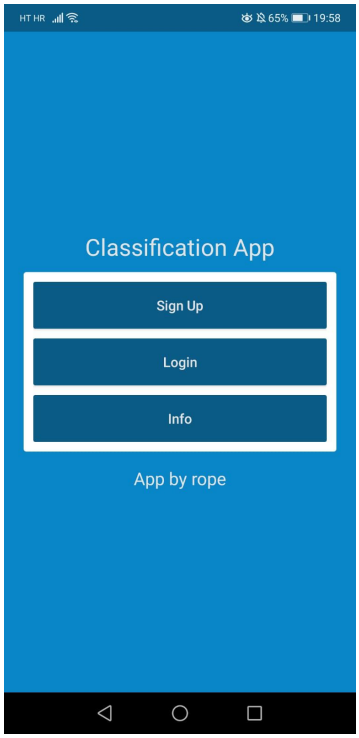
Aplikacija je izgrađena za operacijski sustav Android, uz korištenje programskog jezika Java i biblioteke Retrofit. Android je operacijski sustav otvorenog koda (engl. *open source*) temeljen na Linuxovoj jezgri (engl. *Linux Kernel*), a koristi se u svim vrstama mobilnih uređaja, kao što su pametni telefoni i tableti [17]. Retrofit je HTTP klijentska biblioteka korištena za razvoj Java Android aplikacija koje komuniciraju s REST uslugama na poslužiteljskoj strani [18]. Za potrebe razvoja poslužitelja korišten je razvojni okvir (engl. *framework*) Java Spring i biblioteke Spring Web, Spring Security, PostgreSQL Driver i Spring Data JPA. Spring je aplikacijski razvojni okvir za Java platformu, nudi mnoštvo biblioteka koje služe kao pomoć u ostvarivanju potrebnih funkcionalnosti aplikacija. Spring Web se koristi za izgradnju Spring REST poslužitelja. Spring Security pomaže u implementaciji kontrole pristupa i autentifikacije za Spring poslužitelje. PostgreSQL Driver omogućava Spring aplikacijama povezivanje i uporabu PostgreSQL baze podataka. Spring Data JPA olakšava pristup i komunikaciju s bazom podataka koristeći Java kod [19]. Kao baza podataka korišten je PostgreSQL. To je sustav za upravljanje relacijskim bazama podataka (engl. *rational database management system*) otvorenog koda [20].
Programski alati korišteni za razvoj aplikacije su:

- Android Studio – klijentska strana
- Intellij IDEA – serverska strana
- pgAdmin – baza podataka
- Visual Studio Code – klasifikacija slika korištenjem modela duboke neuronske mreže

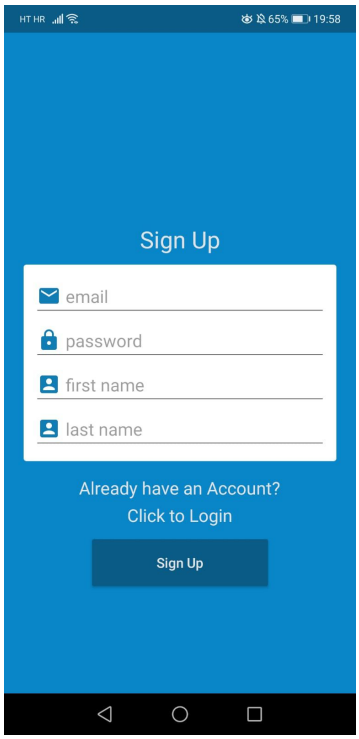
## 5.4. Klijentska strana

Klijentsku stranu sustava čini Android mobilna aplikacija. Aplikacija pruža intuitivno korisničko sučelje i ostvaruje korisnu primjenu razvijenog modela duboke neuronske mreže. U Android okruženju jedan "ekran" aplikacije je aktivnost (engl. *Activity*) i sastoji se od dva dijela: dizajna i funkcionalne logike. Dizajn i izgled korisničkog sučelja ostvaren je XML datotekama razmještaja, dok je za funkcionalnost sučelja zadužena pripadajuća datoteka s Java kodom. Razvijene aktivnosti s pripadajućim funkcionalnostima prikazane su u nastavku.

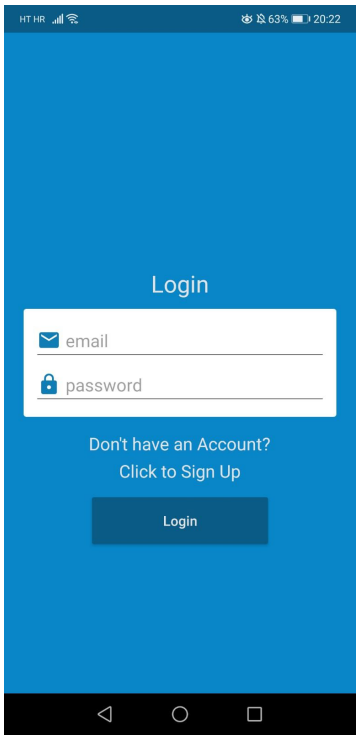
Prilikom pokretanja aplikacije otvara se aktivnost s početnim izbornikom (slika 5.2). Kako je preduvjet za korištenje funkcionalnosti klasifikacije registracija i prijava u sustav, ovdje se korisniku nude opcije prelaska na te aktivnosti.



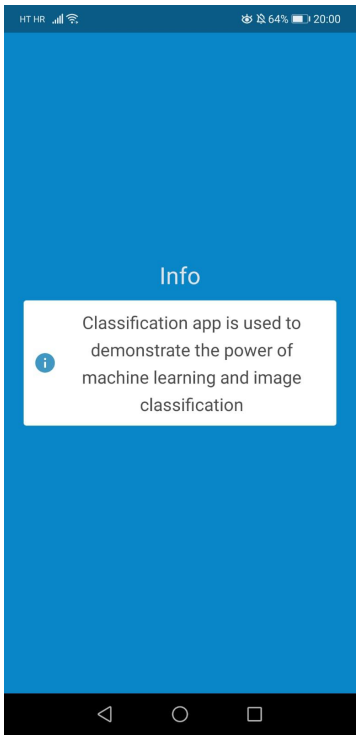

**Slika 5.2:** Početni izbornik




**Slika 5.3:** Registracija




**Slika 5.4:** Login

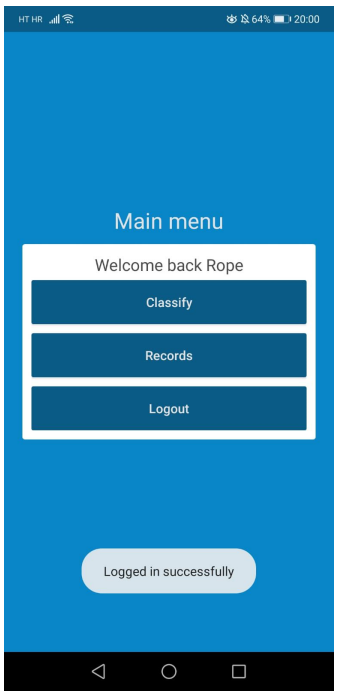



**Slika 5.5:** Info

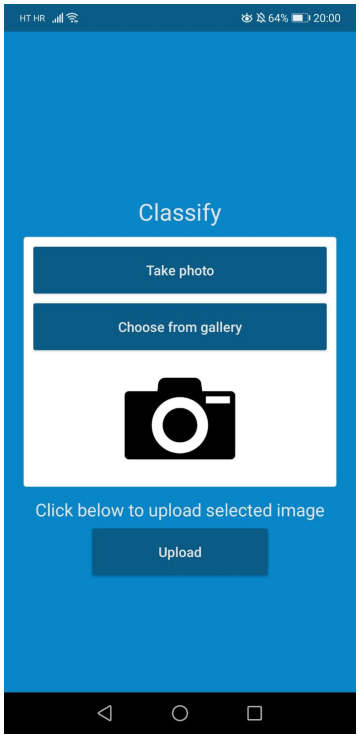
Za registraciju je u "Sign up" formu (slika 5.3) potrebno upisati email adresu, lozinku, ime i prezime. Klikom na gumb "Sign up" pokreće se validacija unesenih podataka. Provjerava se ispravan format email adrese i minimalna duljina lozinke, te postojanje unosa u poljima za ime i prezime. Nakon validacije uneseni podaci šalju se na poslužitelj korištenjem zahtjeva Retrofit POST. Ako neka od validacija ne prolazi podaci se ne šalju na poslužitelj i korisnika se upozorava na pogrešku.

Ako je registracija bila uspješna, moguće se prijaviti u sustav korištenjem forme za prijavu (slika 5.4) u koju je potrebno upisati postojeći email i lozinku. Klikom na gumb "Login" pokreće se već spomenuta validacija za email i lozinku. Nakon validacije uneseni podaci šalju se na poslužitelj korištenjem zahtjeva Retrofit POST na koji poslužitelj odgovara statusom 200 OK u kojem uz osnovne korisničke podatke vraća kolačić za sjednicu (engl. *session cookie*). Korisnički podaci koje poslužitelj vraća u odgovoru na prijavu spremaju se u tzv. $SharedPreferences$ koji služi kao kontejner podataka za sve aktivnosti Android aplikacije, kako bi u svakoj mogli saznati podatak o trenutno prijavljenom korisniku.

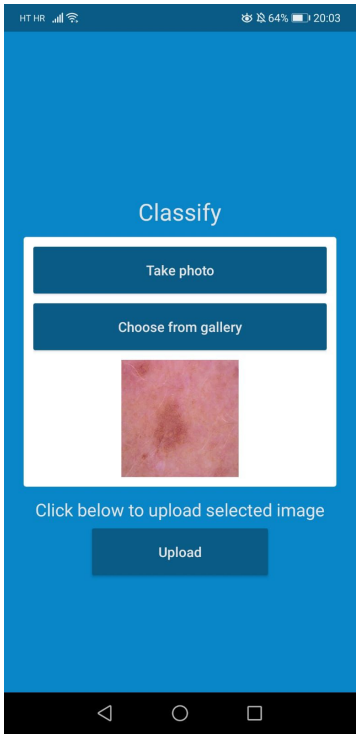
Info aktivnost (slika 5.5) služi za informiranje korisnika o aplikaciji, a mogu joj pristupiti i neprijavljeni korisnici.
Nakon uspješne prijave u sustav uz skočnu poruku prikazuje se glavni izbornik (slika 5.6) s opcijama prelaska na aktivnosti za klasifikaciju ("Classify"), pregled obavljenih klasifikacija ("Records") i odjavu iz sustava ("Logout").



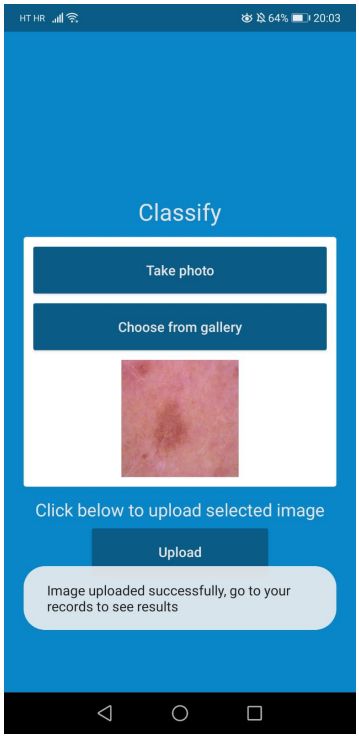

**Slika 5.6:** Glavni izbornik



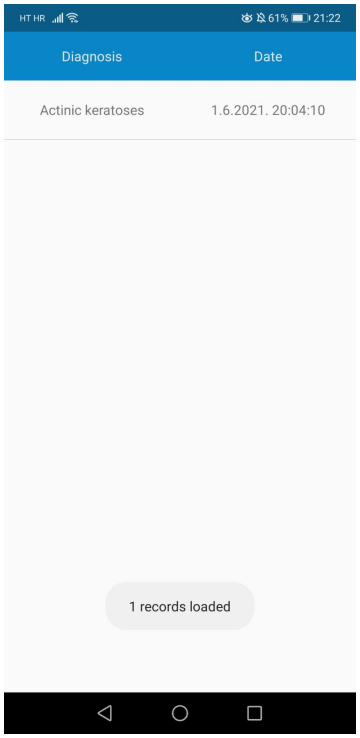

**Slika 5.7:** Aktivnost za klasifikaciju




**Slika 5.8:** Odabir slike




**Slika 5.9:** Skočna poruka




**Slika 5.10:** Rezultati

Na slici 5.7 prikazana je aktivnost za klasifikaciju. U njoj je potrebno priložiti sliku koja će se slati poslužitelju. Aktivnost omogućava korištenje kamere za slikanje kožnog oboljenja ili odabir postojeće slike iz galerije. Ako se aplikacija koristi po prvi put, od korisnika će biti zatraženo dopuštenje za pristup kameri i galeriji pri odabiru tih aktivnosti. U slučaju da korisnik aplikaciji ne dopusti pristup tim resursima, aplikacija će se samo vratiti na prethodnu aktivnost. Nakon odabira slike ona se prikazuje umjetno $placeholder$ ikone fotoaparata i spremna je za slanje na poslužitelj, što je prikazano na slici 5.8. Konačno, klikom na gumb "Upload" odabrana slika šalje se na poslužitelj korištenjem zahtjeva Retrofit POST. Za vrijeme čekanja odgovora s poslužitelja korisniku se prikazuje $progress\ bar$, a nakon uspješnog slanja i skočna poruka prikazana na slici 5.9. Također je osigurano da korisnik ne može slati zahtjev poslužitelju ako prethodno nije odabrao sliku, a ako dođe do greške prilikom klasifikacije na serveru, ispisuje se poruka o pogrešci ili timeouta u slučaju čekanja dužem od 10 sekundi.

Kako bi se prikazali rezultati klasifikacije potrebno je odabrati aktivnost "Records", prikazanu na slici 5.10. U toj aktivnosti prikazan je popis svih obavljenih klasifikacija uz odgovarajuće dijagnoze kožnih oboljenja i vrijeme slanja slike na poslužitelj.

## 5.5. Poslužiteljska strana

Poslužiteljsku stranu sustava čini Springov REST poslužitelj. Taj poslužitelj prima klijentske HTTP zahtjeve na temelju kojih obavlja upite prema bazi podataka i kao povratnu vrijednost klijenti u HTTP odgovoru vraća trenutno stanje traženih resursa [7].

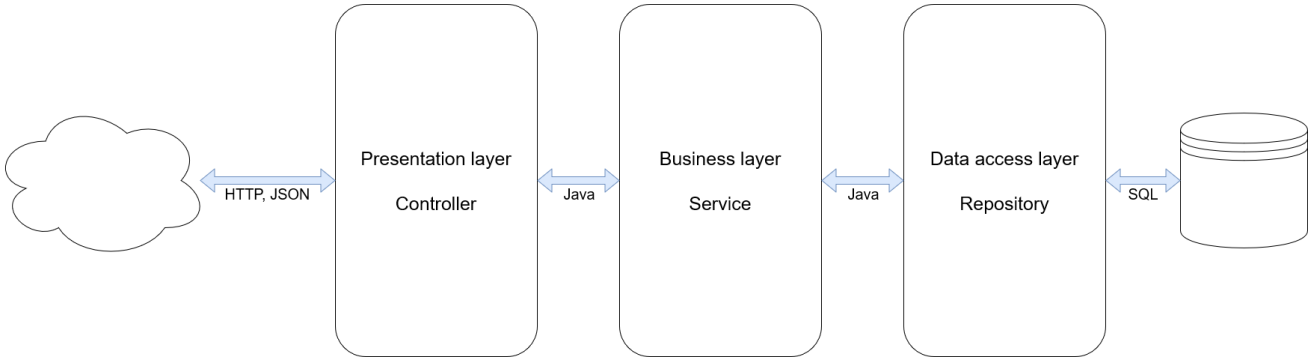
Spring REST poslužitelji izgrađeni su u troslojnoj arhitekturi (engl. *three tier architecture*). Ta arhitektura raspoređuje odgovornosti u aplikaciji na tri dijela: prezentacijski sloj (engl. *presentation layer*), sloj usluge (engl. *business layer*) i podatkovni sloj (engl. *data access layer*). Shematski prikaz takve troslojne arhitekture dan je na slici 5.11.

U prezentacijskom sloju nalaze se REST kontroleri koji prihvaćaju HTTP zahtjeve na određenim putanjama, uz deserijalizaciju podataka poslanih u JSON formatu. Dodatno, s obzirom na korištenje biblioteke Spring Security, uz svaki zahtjev na zaštićenu putanju potrebno je poslati podatke za autentifikaciju. Ako je autentifikacija uspjela nastavlja se obrada zahtjeva, a u suprotnom se klijentu vraća statusni kod "401 Unauthorized" i prekida se daljnja obrada zahtjeva. Prihvaćeni zahtjev se, ovisno o putanji na koju je poslan, prosljeđuje sloju usluge na obradu.

Na sloju usluge nalazi se kod koji odgovara poslovnoj logici sustava, ovdje se

još jednom obavlja validacija podataka nalik onoj na klijentskoj strani. Sloj usluge ima pristup podatkovnom sloju, koji služi kao veza između aplikacije i baze podataka. Nakon obrade proslijeđenog HTTP zahtjeva, sloj usluge ga prosljeđuje podatkovnom sloju.

U podatkovnom sloju se proslijeđeni zahtjev prevodi u SQL upit prema bazi podataka korištenjem biblioteke Spring Data JPA. Kada baza podataka vrati izlazni status s odgovorom, on se prosljeđuje od podatkovnog sloja natrag preko sloja usluge i prezentacijskog sloja pa sve do klijenta.




**Slika 5.11:** Troslojna arhitektura

Na razvijenom REST poslužitelju bitno je naglasiti sljedeće dijelove troslojne arhitekture, uz kratak opis funkcionalnosti.
Prezentacijski sloj:

- RegisterController – prihvat zahtjeva za registraciju
- LoginController – prihvat zahtjeva za login
- RecordController – prihvat zahtjeva sa slikama za klasifikaciju

Sloj usluge:

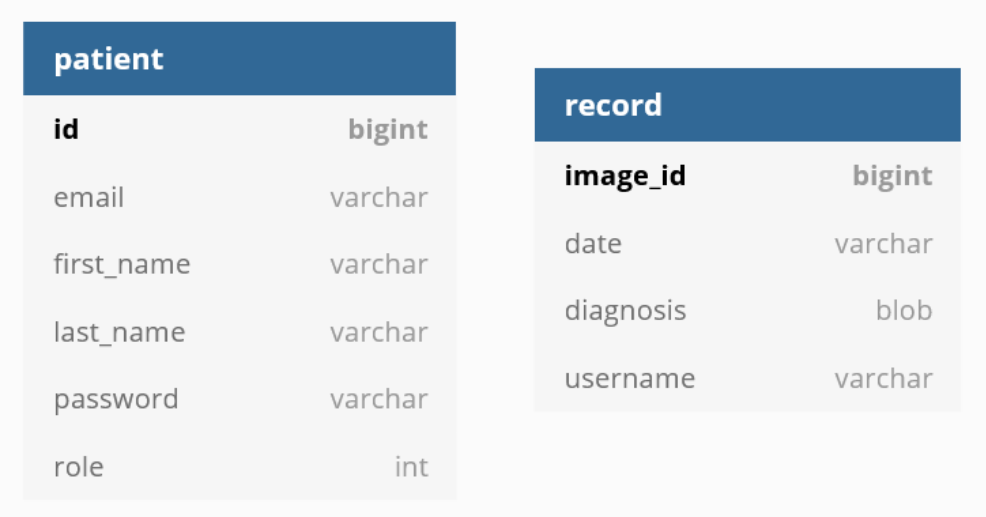
- PatientServiceJpa – logika za stvaranje i traženje korisnika u bazi podataka
- RecordServiceJpa – logika za stvaranje i traženje klasifikacija slika kožnih oboljenja u bazi podataka, ovdje se pokreće skripta s Python kodom u kojem se razvijenim modelom duboke neuronske mreže obavlja klasifikacije poslane slike

Podatkovni sloj:

- PatientRepository – pristup podacima o korisnicima
- RecordRepository – pristup podacima o obavljenim klasifikacijama

## 5.6. Baza podataka

Dio zadužen za bazu podataka u razvijenom sustavu čini postgreSQL, koji osim pohrane pruža i rukovanje podacima u relacijskoj bazi podataka. Bazi pristupamo preko podatkovnog sloja poslužiteljske strane sustava. Dijagram tablica u bazi podataka s pripadajućim tipovima podataka po pojedinim stupcima dani su na slici 5.12.




**Slika 5.12:** Dijagram baze podataka

U tablici "patient" spremljeni su podaci o korisnicima, a podaci o klasifikacijama njihovih kožnih lezija nalaze se u tablici "record". Važno je napomenuti da su tipovi podataka po stupcima točno definirani i provjeravaju se prilikom obrade SQL upita nad bazom podataka. Tako se i u ovom dijelu sustava obavlja validacija podataka, uz spomenute validacije na klijentskoj i poslužiteljskoj strani. Ako dođe do neslaganja između očekivanih i poslanih tipova podataka baza javlja grešku i zaustavlja daljnje izvođenje upita.

# 6. Zaključak

Razvijeni model duboke neuronske mreže sa točnošću 77.0574% klasificira kožne lezije na skupu podataka za testiranje, a mobilna aplikacija koja ga koristi ispunjava postavljene funkcionalne zahtjeve. Aplikacija je osmišljena kao demonstracija praktične primjene modela dubokog učenja. Unaprijeđena verzija ove aplikacije koja bi mogla s većom točnošću klasificirati više od 7 klasa kožnih oboljenja imala bi potencijal za praktičnu uporabu. Takva aplikacija mogla bi se razvijati u smjeru pomoćnog alata u dermatološkoj dijagnostici, no nikako kao zamjena za medicinske stručnjake.

Radom na izradi ovakvog sustava stečena su iskustva u izgradnji Keras modela dubokog učenja, razvoja aplikacija za operacijski sustav Android i implementaciji Spring REST poslužitelja.

# 7. literatura

[1] Porcelain Signapore, 5 Fun Facts About Skin, 3.11.2011., https://porcelainskin.com/prologue/5-fun-facts-about-your-skin/, 20.5.2021.

[2] Artem Oppermann, What is Deep Learning and How does it work?, 12.11.2019., https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac, 20.5.2021.

[3] Gavril Ognjanovski, Everything you need to know about Neural Networks and Backpropagation — Machine Learning Easy and Fun, 14.1.2019., https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a, 20.5.2021.

[4] Buduma, N. Fundamentals of Deep Learning: Convolutional Neural Networks. 1. izdanje. SAD: O'Reilly, 2017.

[5] Sumit Saha, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, 15.12.2018., https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 24.5.2021.

[6] Juliana Negrini de Araujo, HAM10000: Analysis and Model Comparison, 3.1.2021., https://www.kaggle.com/jnegrini/ham10000-analysis-and-model-comparison, 28.5.2021.

[7] Aveek Das, Create REST APIs in Python using Flask, 12.3.2021., https://www.sqlshack.com/create-rest-apis-in-python-using-flask/, 29.5.2021.

[8] Chris Nicholson, Artificial Intelligence (AI) vs. Machine Learning vs. Deep Learning, https://wiki.pathmind.com/ai-vs-machine-learning-vs-deep-learning, 20.5.2021.

[9] Marina Chatterjee, Top 20 Applications of Deep Learning in 2021 Across Industries, 19.2.2019., https://www.mygreatlearning.com/blog/deep-learning-applications/, 20.5.2021.

[10] Markus Schmitt, Artificial Intelligence in Medicine, https://www.datarevenue.com/en-blog/artificial-intelligence-in-medicine, 20.5.2021.

[11] Šitum, M. Dermatovenerologija. 1. izdanje. Zagreb: Medicinska naklada, 2018.

[12] Harvard Dataverse, The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions, https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T, 22.5.2021.

[13] Arc, Convolutional Neural Network, 25.12.2018., https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05, 25.5.2021.

[14] François Chollet et al. Keras. https://keras.io, 2015., 5.6.2021

[15] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Vir-tanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J.Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matt-hew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson,Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, HameerAbbasi, Christoph Gohlke, i Travis E. Oliphant. Array programming with NumPy.Nature, 585(7825):357–362, Rujan 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2, 5.6.2021.

[16] The pandas development team. pandas-dev/pandas: Pandas, Velja˘ca 2020. URL https://doi.org/10.5281/zenodo.3509134.Chollet et al. (2015) Harris et al. (2020) pandas development team (2020), 5.6.2021.

[17] Android, https://www.android.com/, 6.6.2021.

[18] Retrofit, https://square.github.io/retrofit/, 6.6.2021.

[19] Spring, https://spring.io/, 6.6.2021.

[20] PostgreSQL, https://www.postgresql.org/, 5.6.2021.

[21] Kaggle, https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000/code, 7.6.2021.

**Mobilna aplikacija za klasifikaciju kožnih oboljenja iz slika kože**

**Sažetak**

Ovim radom demonstrirana je praktična primjena modela dubokog učenja. Prikazan je način izgradnje modela za klasifikaciju kožnih oboljenja iz slika kože i mobilne aplikacije koja ga koristi.

**Ključne riječi:** Duboko učenje, duboke neuronske mreže, Keras, klasifikacija, Android, Spring, postgreSQL, kožna oboljenja, mobilna aplikacija

**Mobile application for the classification of skin diseases from skin images**

**Abstract**

This paper demonstrates the practical application of the deep learning model. The method of building a model for the classification of skin diseases from skin images and the mobile application that uses it is presented.

**Keywords:** Deep learning, deep neural networks, Keras, classification, Android, Spring, postgreSQL, skin diseases, mobile application