Prepoznavanje simbolov s pomočjo konvolucijskih nevronskih mrež

Računalništvo in informatika

Raziskovalna naloga

Avtor: Jerman David, G 4. B

Mentor: Volčini Aleš, prof.

Vegova Ljubljana

2021

Kazalo vsebine

Kazalo slik

Povzetek

Cilj raziskovalne naloge je bila izdelava modela konvolucijske nevronske mreže (ang. *convolutional neural network,* CNN), po katerem bi bilo možno razlikovati med 62 različnimi simboli angleške abecede in številkami od 0 do 9. Ugotavljal sem tudi, ali je CNN res primerna vrsta umetnih nevronskih mrež za tovrstno nalogo. Odločil sem se tudi, da izdelam program z grafičnim vmesnikom za izdelavo modelov CNN, ki pa mi je olajšal izdelavo, treniranje in testiranje modela CNN. Celoten projekt je bil izdelan s pomočjo programskega jezika python in popularne knjižnice, orodja za strojno učenje Keras, ki pa jo uporabljajo tudi znana podjetja kot je Google.

V okviru projekta sem tako izdelal dva programa z grafičnim vmesnikom za treniranje in izdelavo modela, ter program za testiranje modela. S pomočjo programa sem nato izdelal preprost model CNN iz več slojev, ki pa je bil po treniranju sposoben razlikovati med 62 različnimi ročno narisanimi simboli. Model sem tudi testiral in ugotovil, da ima le-ta težave pri razlikovanju nekaterih simbolov, kot sta črka o in številka 0.

Ključne besede: *strojno učenje, umetne nevronske mreže, CNN*

Summary

The aim of my research was to create a model of convolution neural network (CNN) capable of distinguishing between 62 different symbols of English alphabet and digits from 0 to 9. I also tried to figure out, whether CNN is the right type of artificial neural networks for this kind of a task. In addition to that, I decided to make a program with a graphical interface for building CNN models, which made the creation, training and testing of the CNN model easier. The entire project was created using the Python programming language and the popular machine learning library Keras, which is also used by well-known companies such as Google.

As part of the project, I have created two programs with a graphical interface: for training and building a model, and for testing the model. With the help of the program, I then created a simple CNN model consisting of several layers, which after training was able to distinguish between 62 different hand-drawn symbols. I also tested the model myself and found that it had difficulty distinguishing some symbols, such as the letter o and the digit 0.

Keywords: *machine learning, artificial neural networks, CNN*

Uvod

Zanimanje za umetno inteligenco in strojno učenje se je ob prihodu nove tehnologije, ki pa je omogočila uporabo le-tega v vsakdanjem življenju, neizmerno povečalo. Strojno učenje računalnikom namreč omogoča opravljanje nalog, za katere bi drugače mislili, da jih lahko opravlja samo človek, kar vključuje prepoznavanje simbolov na slikah, igranje zapletenih iger kot je go itd. Zato sem se tudi sam začel zanimati za strojno učenje in umetno inteligenco. Ker sem znanje programskega jezika python imel že od prej, sem se odločil, da se tudi sam preizkusim na tem področju. Tako sem se namenil izdelati model CNN, ki bo imel sposobnost razločiti med 62 različnimi simboli abecede in števkami od 0 do 9.

Hipoteze

V raziskovalni nalogi obravnavam naslednje hipoteze:

1. Predpostavljam, da je mogoče izdelati preprost model, ki ima s pomočjo strojnega učenja zmožnost z zadovoljivo natančnostjo (vsaj 90%) prepoznavati simbole na slikah.
2. Predpostavljam, da se za prepoznavanje simbolov na slikah konvolucijska nevronska mreža obnese zelo dobro.
3. Predpostavljam, da je preprost program z grafičnim vmesnikom modelov CNN mogoče izdelati le z uporabo programskega jezika python in dveh knjižnic, Tkinter in Keras.

Strojno učenje in umetne nevronske mreže

Ker je strojno učenje precej nova tematika v svetu računalništva, menim da je za razumevanje raziskovalne naloge potrebno vsaj osnovno znanje o strojnem učenju in umetnih nevronskih mrežah.

Strojno učenje

Strojno učenje je znanstvena veda, ki se ukvarja z razvojem algoritmov, statističnih modelov in računalniških sistemov, ki imajo zmožnost opravljanja raznih nalog brez eksplicitnih navodil. Namesto tega se tovrstni algoritmi in modeli zanašajo na vzorce in sklepanja. Matematični modeli bazirajo na algoritmih strojnega učenja in se opravljanja določene naloge naučijo na podlagi velike količine podatkov (več podatkov model prejme, boljši je pri opravljanju svoje naloge). S pomočjo teh podatkov se torej naučijo dajanja napovedi, kot je npr. napovedovanje cene delnic, in sprejemanja odločitev. Strojno učenje je ravno zaradi dejstva, ker zahteva bistveno manj programiranja za opravljanje nalog kot je prepoznavanje obrazov na slikah, avtonomna vožnja avtomobilov ipd., postalo zelo priljubljeno že v 20. st. Danes je strojno učenje mogoče zaslediti vsepovsod okoli nas. Primer pojavljanja tovrstne tehnologije so filtri v spletni pošti, avtonomna vozila, digitalni asistenti in razne storitve kot je Google prevajalnik. Tudi priporočila za izdelke, ki bi jih morda želeli kupiti, ko kupujemo v spletni trgovini, so rezultat strojnega učenja. Strojno učenje je močno povezano tudi z umetno inteligenco, saj je osnova za le-to ravno strojno učenje.

Vrste strojnega učenja

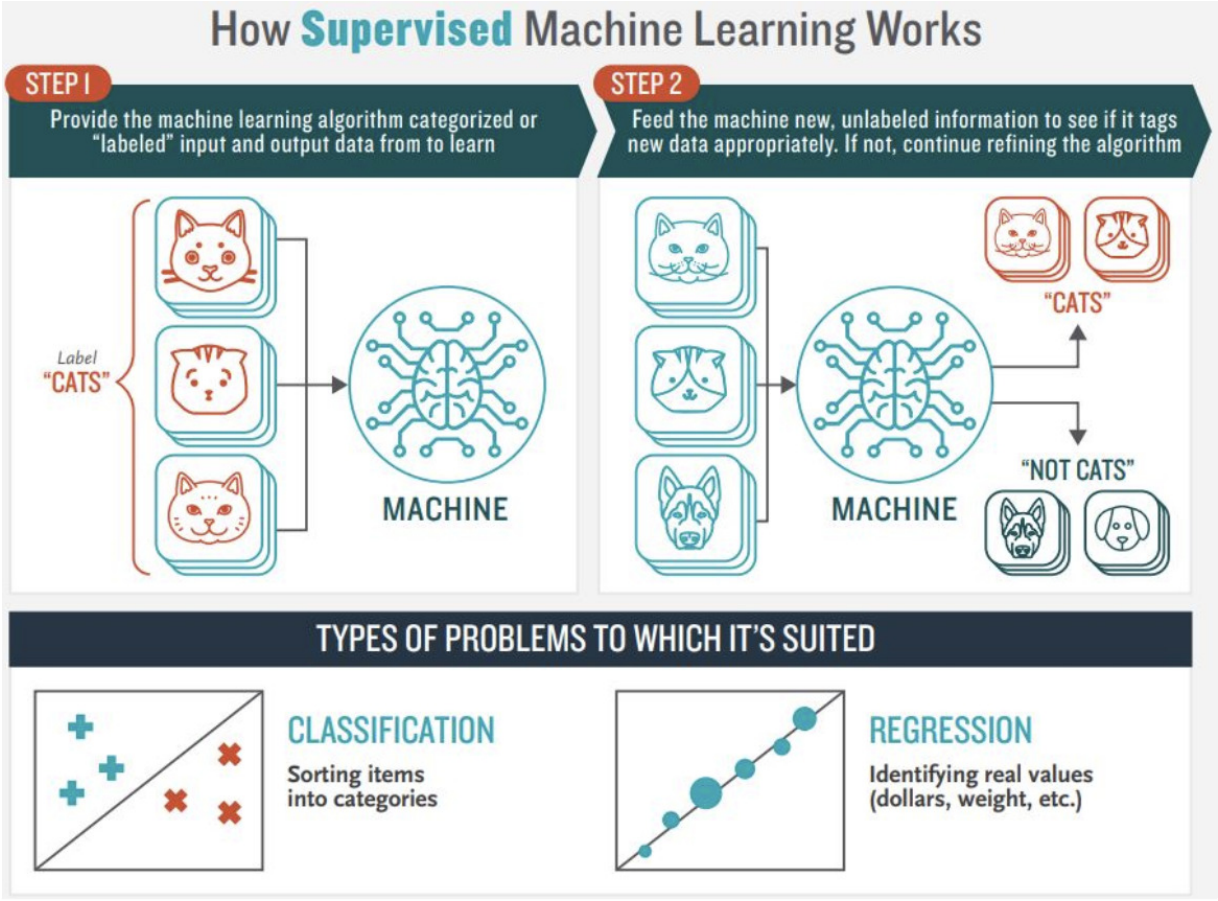
Poznamo več pristopov strojnega učenja, ki pa se med sabo razlikujejo glede na način učenja in glede na vrsto algoritma, ki je uporabljen v določenem matematičnem modelu. Glede na način učenja se strojno učenje deli v tri (štiri) podvrste:

- nadzorovano učenje *(ang. supervised learning),*
- nenadzorovano *(ang. unsupervised learning),*
- delno nadzorovano strojno učenje *(ang. semi-supervised learning)* in
- vzpodbujevalno strojno učenje *(ang. reinforcement machine learning).*

Algoritme strojnega učenja pa lahko delimo tudi glede na njihove podobnosti. Tako jih delimo v naslednje skupine: *regression algorithms, instance-based algorithms, regularization algorithms, decision tree algorithms, Bayesian algorithms, clustering algorithms, association rule algorithms, artificial neural network algorithms, deep learning algorithms, dimensionality reduction algorithms, ensemble algorithms.* Vseh naštetih algoritmov v raziskovalni nalogi ne bom niti podrobno opisoval niti navajal slovenskih prevodov, saj niso vsi tema moje raziskovalne naloge, temveč se bom bolj podrobno posvetil le algoritmom umetnih nevronskih mrež (ang. *artificial neural network algorithms*). Algoritme umetnih nevronskih mrež sem namreč tudi sam uporabil v svojem projektu.

Nadzorovano strojno učenje

Nadzorovano strojno učenje je prvi izmed pristopov strojnega učenja. To je način strojnega učenja s pomočjo podatkov, ki pa so urejeni. Podatke, ki jih algoritem strojnega učenja prejme v procesu učenja imenujemo učna podatkovna množica (ang. *training data*



Slika 1: Nadzorovano strojno učenje

set). Ti podatki so urejeni tako, da ima vsak podatek tudi oznako, ki pove, kakšna bi morala biti pravilna napoved algoritma po prejetju tega podatka. V procesu učenja se tako spreminjajo lastnosti algoritma s ciljem, da bi lahko algoritem vrnil čim bolj pravilne napovedi ob prejetju podatkov, ki pa ne bodo označeni in urejeni.

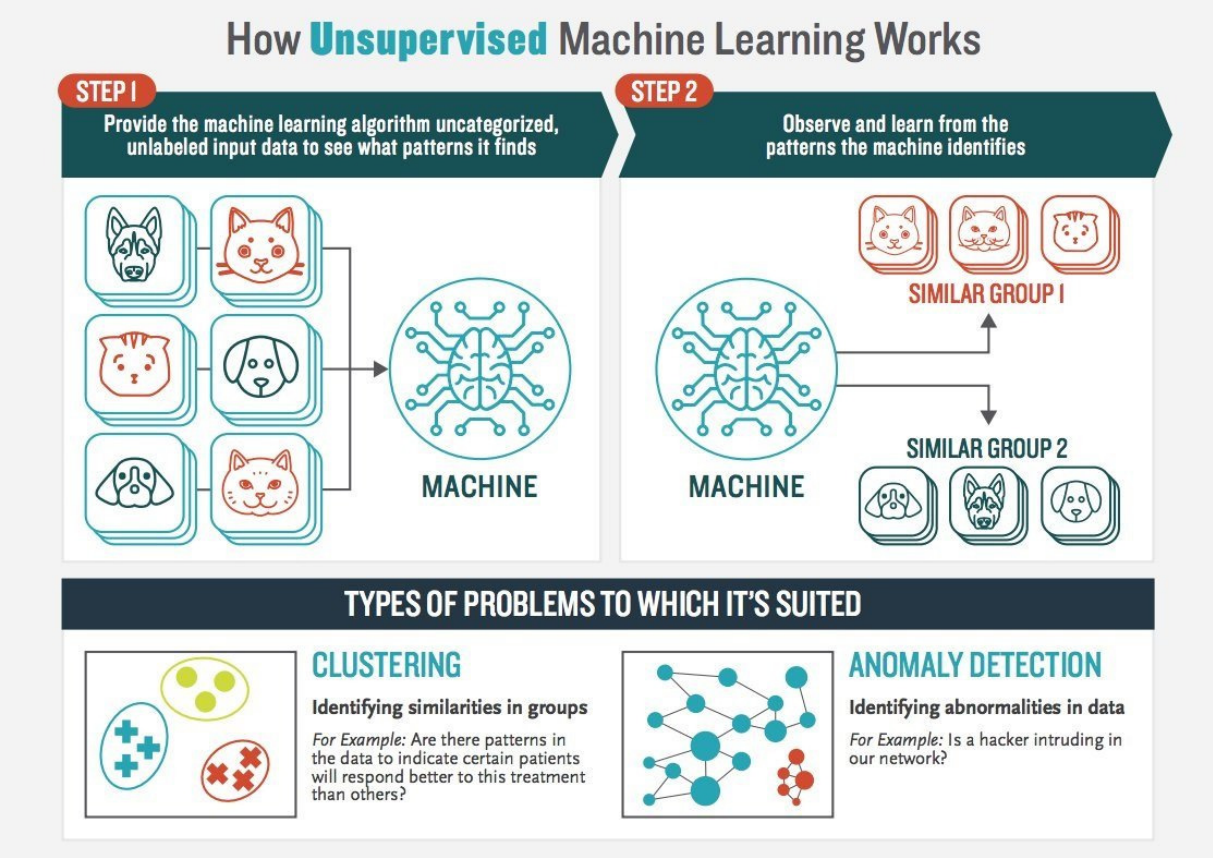
Za primer vzemimo za podatke slike mačk in psov. Slike morajo biti v procesu učenja urejene, torej ločiti moramo slike psov od slik mačk tako, da npr. slike mačk poimenujemo drugače od slik psov ali pa jih ločimo tako, da jih damo v različne direktorije. Algoritem strojnega učenja nato sprejme te slike in se poskuša naučiti razlikovati med slikami psov in mačk. Po končanem procesu učenja naj bi imel algoritem tako zmožnost razločiti med sliko mačke in sliko psa brez da bi mu povedali, kaj dejansko na sliki je. Opisani primer je prikazan tudi na sliki na prejšnji strani.

Algoritmi strojnega učenja, ki pa uporabljajo ta način strojnega učenja pa so klasifikacija, regresija (logistična in linearna), umetne nevronske mreže, SVM itd.

Nenadzorovano strojno učenje

Nenadzorovano strojno učenje je način strojnega učenja s pomočjo podatkov, ki pa za razliko od podatkov nadzorovanega načina učenja niso urejeni, niti niso označeni. Učenje zato v tem primeru ne poteka na enak način. Ker podatki niso urejeni, cilj algoritma torej ni ustvarjati napoved, temveč podatke na nek način razvrstiti (glede na podobnosti med njimi, v skupine itd.). Cilj tovrstnega algoritma je lahko tudi odkriti razna pravila povezanosti med podatki, torej na kakšen način ali po kakšnem vzorcu so ti že povezani med sabo. Tovrstni algoritmi lahko v skupini podatkov iščejo tudi anomalije oz. odstopanja od ostalih podatkov (npr. vsiljivce, kot so rakave celice ipd.).

Primer takšnega načina strojnega učenja je lahko ravno tako razvrščanje slik psov in mačk, le s to razliko da se model ne nauči razlikovati mačk in psov s pomočjo urejenih podatkov in učenja, temveč slike loči le na podlagi podobnosti slik med sabo, a v tem primeru se algoritem ne zaveda, da ločuje med mačkami in psi, saj tega ni bil naučen, ve le da slike ločuje v skupine na podlagi podrobnosti s slik. Opisani primer je prikazan tudi na sliki na naslednji strani.



Slika 2: Nenadzorovano strojno učenje

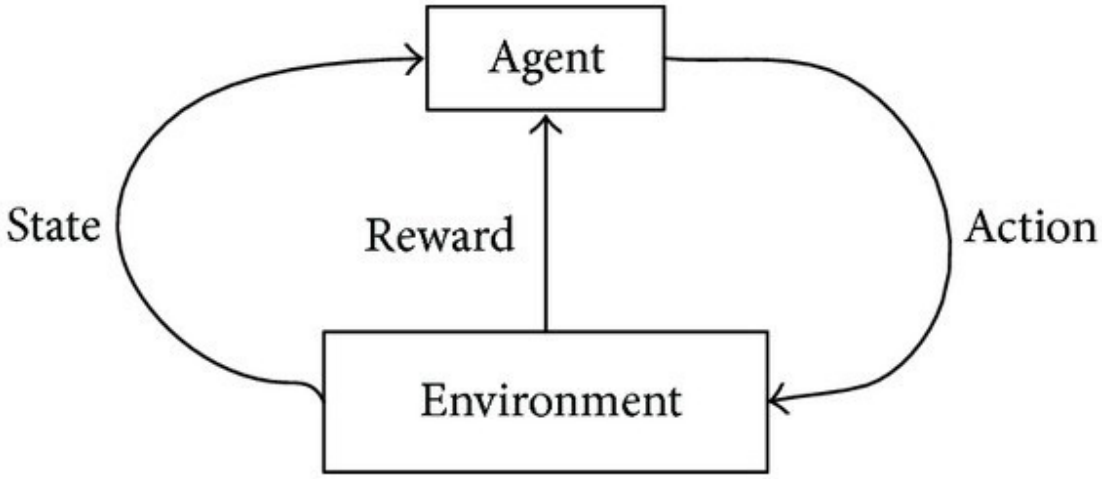
V to skupino algoritmov spadajo pristopi kot sta algoritma *Apriori* in *K-Means* (*clustering*).

Delno nadzorovano strojno učenje

Delno nadzorovano strojno učenje je mešanica prej omenjenih načinov strojnega učenja, nadzorovanega in nenadzorovanega strojnega učenja, saj so podatki, ki jih algoritem prejme delno urejeni in delno neurejeni. Algoritem tako poskuša najti povezanosti med podatki ali pa le-te razvrstiti, poleg tega pa se algoritem uči tudi s pomočjo podatkov, ki pa so urejeni in označeni.

Vzpodbujevalno strojno učenje

Vzpodbujevalno strojno učenje je način strojnega učenja, ki ga je mogoče razložiti s spodnjim diagramom. Agent (naš model strojnega učenja) se opravljanja naloge nauči s pomočjo poskušanja in napak (ang. *trial and error*). Za lažjo razlago za primer vzemimo model strojnega učenja, ki se poskuša naučiti z lokom zadeti tarčo. Učenje se prične s tem, da model nekaj stori, v našem primeru izstreli puščico. Če puščica uspešno zadene tarčo, potem je model nagrajen in tako se po več uspešnih poskusih nauči, kako zadeti tarčo. Kadar model zgreši tarčo, pa ni nagrajen, ampak kaznovan. Cilj modela je tako zbrati čim več nagrad, ter se tako posledično naučiti opravljanja določene naloge. Ta način strojnega učenja je dejansko enak človekovemu načinu učenja, saj smo tudi ljudje kot otroci za pravilna dejanja nagrajeni, za napačna pa kaznovani. Takšen način učenja je zelo pogosto uporabljen v razvoju raznih programov, ki se naučijo igranja namiznih iger kot sta šah in go. Primer tovrstnega programa je Googlov AlphaGo, ki pa se je naučil igranja igre go ter uspel premagati celo svetovnega prvaka v igranju go.



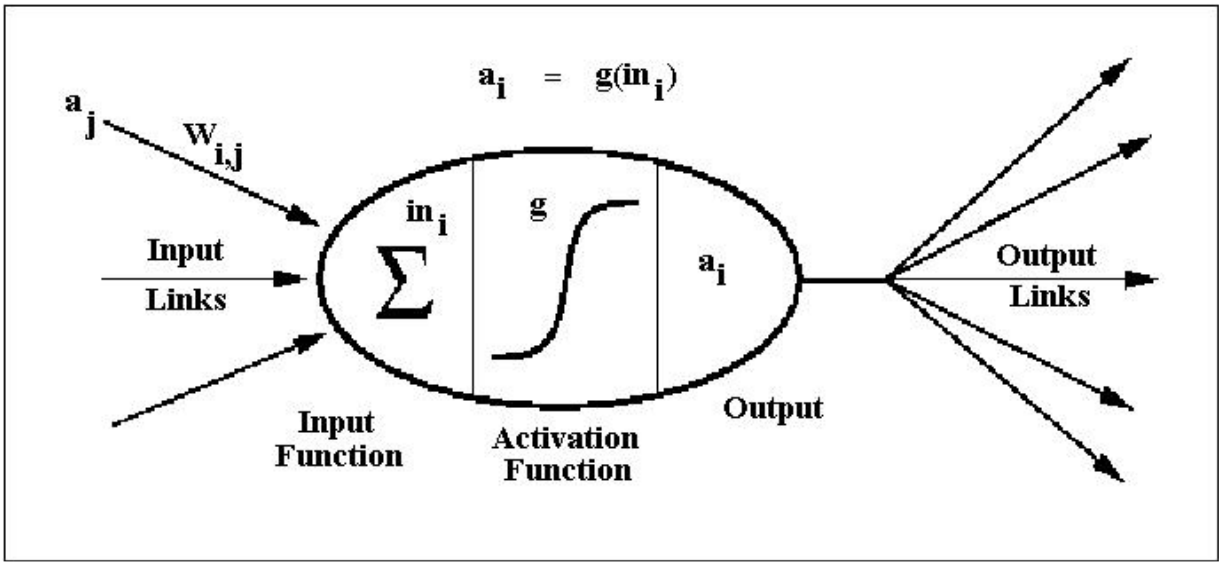
Slika 3: Vzpodbujevalno strojno učenje

Umetne nevronske mreže

Umetne nevronske mreže so eden izmed pristopov strojnega učenja in so nastale po vzoru delovanja nevronov v naših možganih. Tovrstni algoritmi spadajo v skupino algoritmov nadzorovanega strojnega učenja.

Zgradba umetnih nevronskih mrež

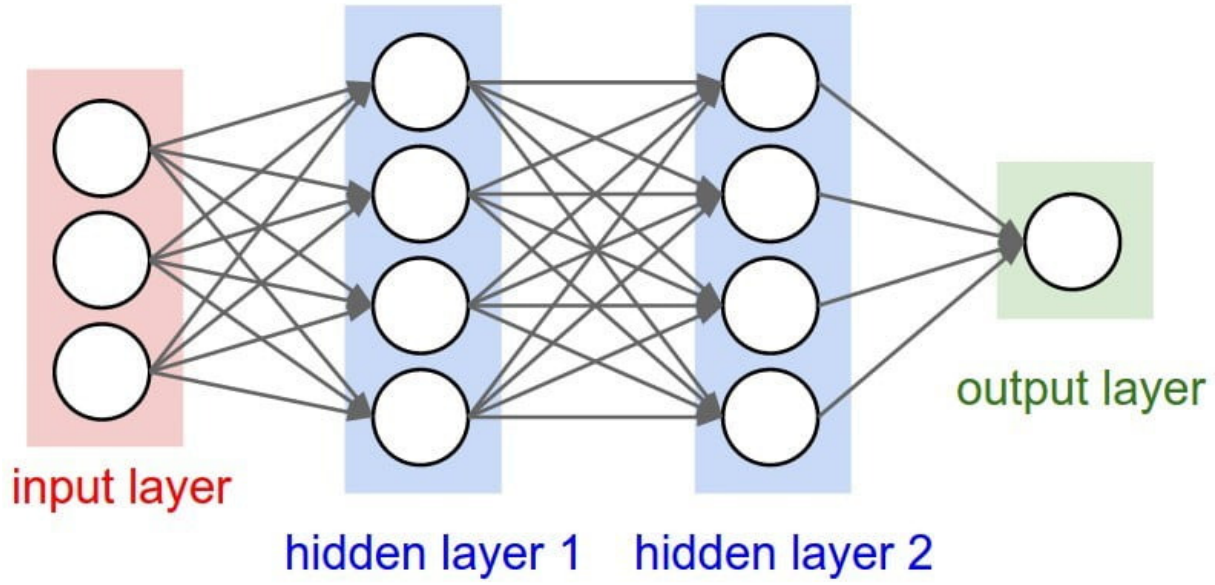
Osnovni gradnik vsake nevronske mreže je nevron. Nevron umetne nevronske mreže ima enako funkcionalnost kot nevroni v naših možganih. Nevron prejme več signalov ter le-te sešteje s pomočjo vhodne (ang. *propagation* oz. *input*) funkcije, ki pa svoj rezultat posreduje aktivacijski funkciji (ang. *activation function*). Aktivacijska funkcija odloča o tem, ali se bo v nevronu sprožil signal ali ne. Ta signal postane tudi izhodna funkcija nevrona (ang. *output function*). Nevroni so med seboj povezani, in sicer je izhod nevrona običajno povezan z več vhodi nevronov na naslednjem sloju.



Slika 4: Zgradba umetnega nevrona

Vsaki povezavi med nevroni pa je prirejena tudi utež (ang. *weight*). Utež predstavlja pomembnost nevrona oz. povezave med njima. Bolj pomembna kot je povezava med nevronoma, večja je utež in večji vpliv ima nevron na končni rezultat.

Nevroni so običajno povezani v različne sloje. Običajno so iz več slojev sestavljene nevronske mreže globokega učenja (ang. *deep learning neural networks*). V tem primeru je vsak nevron povezan z vsemi nevroni na naslednjem sloju, če gre za polno povezano (ang. *fully connected*) umetno nevronsko mrežo.



Slika 5: Zgradba umetne nevronske mreže

Sloje pa glede na funkcijo, ki jo ti opravljajo v umetni nevronski mreži kot celoti, delimo na tri skupine. Prvi sloj se imenuje vhodni sloj (ang. *input layer*). Naloga tega sloja je sprejemanje podatkov iz okolja, kot so npr. slike ali razne tabele. Naslednji sloj se imenuje skriti sloj (ang. *hidden layer*). V umetni nevronski mreži se po navadi nahaja več skritih slojev (poznamo pa tudi vrste nevronskih mrež brez skritih slojev), ki pa so lahko različnih vrst. Naloga tega sloja je običajno prepoznavanje vzorcev v podatkih posredovanih preko vhodnega sloja. Zadnji sloj v umetni nevronski mreži pa se imenuje izhodni sloj (ang. *output layer*). Ta sloj je odgovoren za dajanje končne napovedi modela, zato je število nevronov na tem sloju enako številu možnih različnih napovedi.

Delovanje in proces učenja umetne nevronske mreže

Modeli umetnih nevronskih mrež se učijo s pomočjo urejenih podatkov, saj kot sem že omenil, umetne nevronske mreže spadajo v pristop nadzorovanega strojnega učenja.

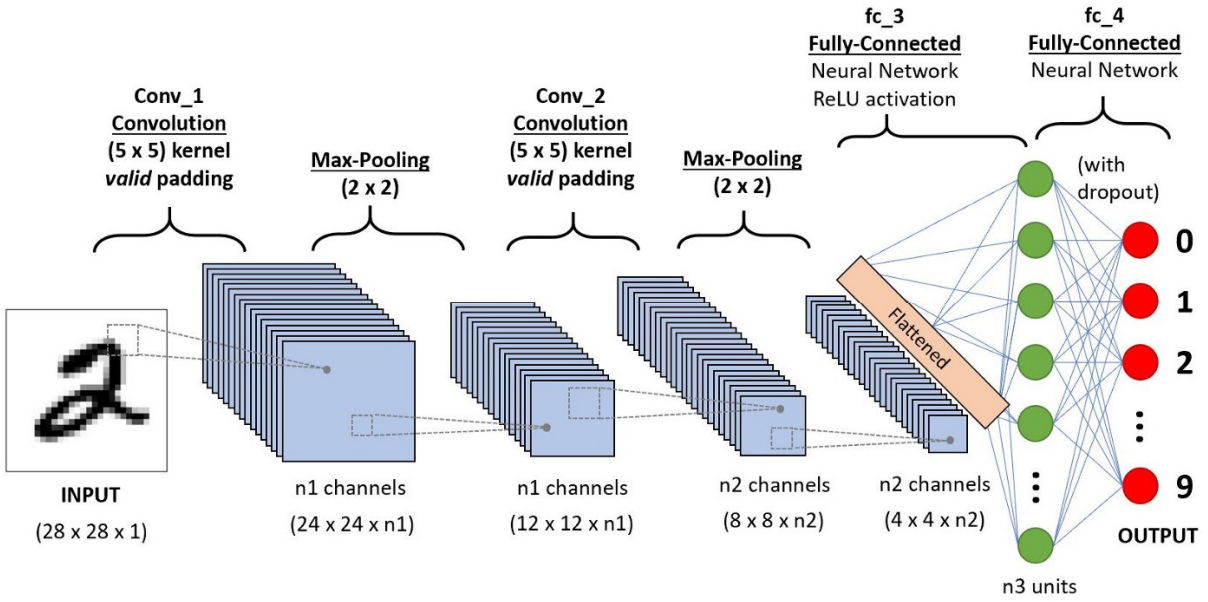
Učenje se prične s tem, da so uteži med nevroni najprej postavljene na neko naključno vrednost. Zatem umetni nevronski mreži posredujemo naše podatke (ang. *forward propagation*), to so lahko slike ali razne tabele z informacijami. S pomočjo danih podatkov model ustvari napoved na podlagi trenutne zgradbe modela, ki pa je naključna, saj so uteži med nevroni naključno določene. Ko model da neko napoved, se le-ta oceni kot pravilna ali napačna (ang. *output error*). Na tej točki se izračuna tudi natančnost modela s pomočjo funkcije izgube (ang. *loss function*), ki pa nam pove kako natančna je bila napoved modela. V primeru da je napoved napačna, se izvedejo popravki na lastnostih modela, torej spremenijo se vrednosti uteži s ciljem, da bi bila napoved modela naslednjič natančnejša. Temu načinu rečemo tudi metoda vzvratnega razširjanja (ang. *backpropagation*). Postopek učenja lahko traja več minut, več ur ali celo več dni, odvisno od zahtevnosti našega modela in moči našega računalnika, izvaja pa se toliko časa, dokler ne dosežemo želene natančnosti modela oz. procesa učenja ne prekinemo.

Konvolucijske nevronske mreže (CNN)

Ena izmed oblik umetnih nevronskih mrež so tudi konvolucijske nevronske mreže (ang. *convolution neural network,* CNN). Ta je najbolj uporabljena na področju računalniškega vida (ang. *computer vision*), zato sem le-to uporabil tudi v moji raziskovalni nalogi.

CNN je eden izmed algoritmov globokega učenja (ang. *deep learning*), ki za vhodne podatke sprejme slike in lahko le-te med seboj razlikuje po tem, kaj se nahaja na sliki. Ker je tudi CNN umetna nevronska mreža, je tudi ta sestavljena iz nevronov, ki se prav tako povezujejo v sloje. CNN ima običajno veliko večjo uspešnost v primerjavi z običajnimi umetnimi nevronskimi mrežami, saj model uporablja razne filtre, ki lahko na sliki prepoznajo določene lastnosti, po katerih eno sliko razločijo od druge. Poleg tega CNN omogoča tudi zmanjšanje parametrov potrebnih za izdelavo modela, kar pa znatno zmanjša čas učenja. CNN to doseže z uporabo več slojev: konvolucijskih slojev, ki služijo za ekstrakcijo pomembnih lastnosti slike, združevalnih (ang. *pooling*) slojev, ki služijo za

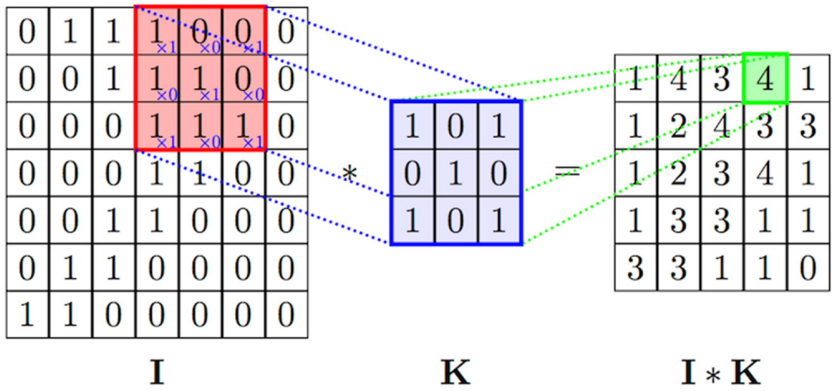
dimenzionalno redukcijo in sploščitvenega (ang. *flatten*) sloja za »sploščitev« prejšnjih slojev, ter običajnih polno povezanih (ang. *dense*) slojev.



Slika 6: Zgradba CNN

Konvolucijski sloj

Konvolucijski sloj deluje po tem principu, da vhodni sloj (torej matrika slike ali prejšnjega sloja) I, pomnoži z nekim filtrom K (ang. *filter* ali *kernel*) (slika 7). Ta postopek imenujemo konvolucija (ang. *convolution*) in le-ta nam omogoča ekstrakcijo določenih lastnosti slike, kot so npr. robovi, kar je



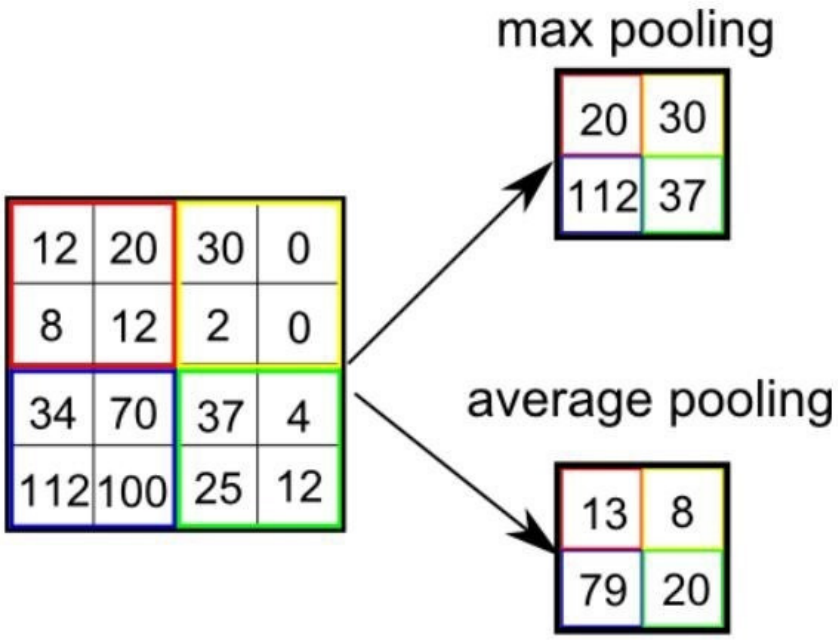
Slika 7: Konvolucija

Slika 8: Ekstrakcija pomembnih lastnosti slike s pomočjo konvolucije

prikazano tudi na sliki 8. Namen konvolucije je v glavnem izluščiti visokonivojske lastnosti vhodne slike, kot so, kot sem že omenil, robovi, barva slike ipd. Vsak konvolucijski sloj je običajno sestavljen iz več različnih filtrov, ki iz slike poskušajo izluščiti različne lastnosti. Matrika, ki jo dobimo ob procesu konvolucije, se imenuje nabor značilk (ang. *feature map*).

Združevalni sloj

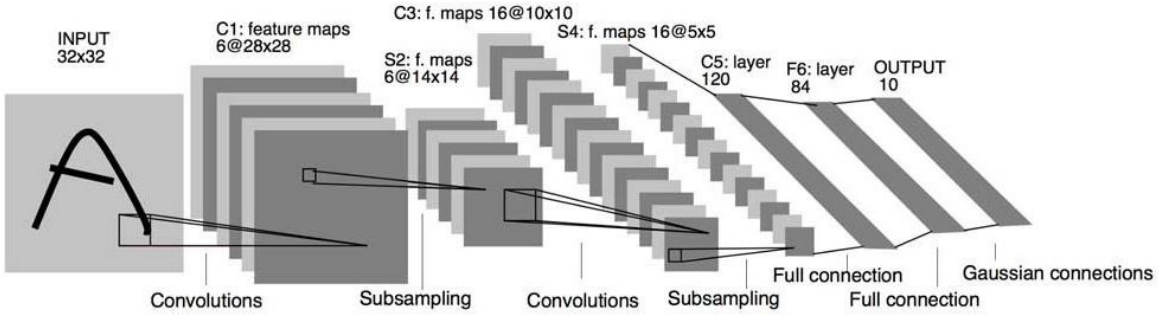
Namen združevalnega sloja je zmanjšanje dimenzij matrik, pridobljenih v procesu konvolucije. S tem namreč znatno zmanjšamo računsko moč potrebno za procesiranje podatkov v modelu CNN. Proces zmanjšanja dimenzij matrik imenujemo dimenzionalna redukcija (ang. *dimensionality reduction*). Združevalni sloji z maksimizacijo (ang. *max pooling*) pa poleg tega omogočajo tudi ekstrakcijo dominantnih lastnosti matrike. Združevanje (ponekod temu pravijo tudi agregacija) podobno kot tudi konvolucija za ekstrakcijo oz. dimenzionalno redukcijo uporablja filter oz. jedro matrike. Razlika pa je v tem, da združevanje v delu matrike, ki ga pokriva filter, vrne največjo ali pa povprečno vrednost jedra matrike odvisno od vrste združevanja. Lahko bi rekli tudi, da združevanje v vsakem »bazenu« matrike vrne neko vrednost glede na vrsto združevanja.



Slika 9: Max pooling in average pooling

Polno povezani sloj

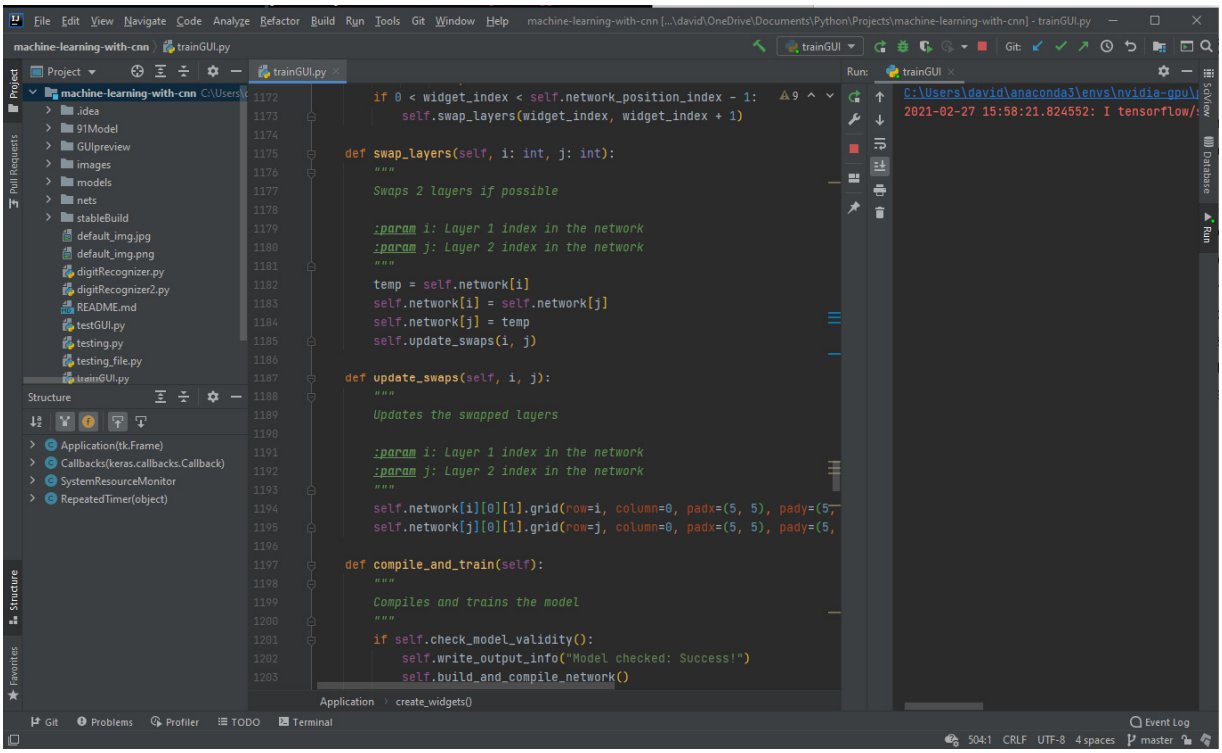
Polno povezani sloji (ang. *fully-connected layer*) so zadnji sloj CNN. Polno povezani sloj sestoji iz sploščitvenega sloja in umetne nevronske mreže z izhodnim slojem. Sploščitveni sloj je zadolžen za to, da »splošči« matrike, ki so nastale v procesu konvolucije in dimenzionalne redukcije. S tem sliko (ali slike) pretvorimo v enodimenzionalni vektor, katerega posredujemo običajni nevronski mreži, ki pa se v procesu učenja nauči prepoznavanja simbolov na slikah ipd. Na spodnji sliki je tako prikazan primer strukture CNN, ki sem je opisal:



Slika 10: Zgradba CNN (LeNet by Yann LeCun)

Prepoznavanje simbolov s pomočjo CNN

Cilj mojega projekta je bil ustvariti preprost program z grafičnim vmesnikom za izdelavo, učenje in preizkušanje modelov CNN. S tem namenom sem izdelal dva programa, enega za treniranje in drugega za testiranje. Oba programa sta izdelana v programskem jeziku python. Zanj sem se odločil, ker je zelo razširjen, dobro dokumentiran in preprost programski jezik. Ena izmed njegovih slabosti je hitrost, a to na kvaliteto mojega projekta ni vplivalo, saj sem v pythonu spisal samo grafični vmesnik. Za zaledni sistem (ang. *backend*) pa sem uporabil knjižnico Tensorflow GPU, ki je spisana v jeziku C++, ta pa nima problemov s hitrostjo. Za izdelavo programov sem uporabil IntelliJ, ki je urejevalnik kode oz. razvojno okolje. Je eden izmed najbolj razširjenih, saj je opremljen z vsemi orodji, ki jih povprečen programer potrebuje za svoje delo. Za lažji nadzor nad stanjem projekta pa sem uporabil tudi Git, saj mi ta omogoča lažje sledenje spremembam projekta.



Slika 11: Urejevalnik kode IntelliJ

Program za izdelavo in treniranje modela CNN

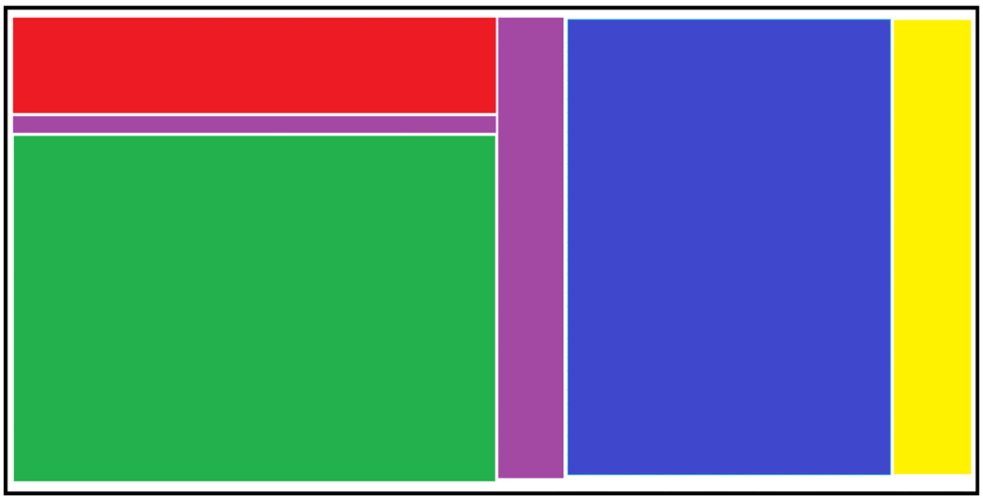
Z namenom, da bi poenostavil proces izdelovanja modelov CNN in treniranje le-teh, sem se odločil, da izdelam program z grafičnim vmesnikom, ki uporabniku omogoča izdelavo poljubnega modela CNN brez potrebe po pisanju kakršnekoli programske kode. To zelo pohitri proces izdelave modela CNN, poleg tega pa je ta program dejansko primeren tudi za uporabnike, ki znanja o programiranju nimajo, a se spoznajo na strojno učenje. Cilj pri razvoju tega programa sta bila torej preprostost in uporabnost. Program sem, kot sem že omenil, izdelal s pomočjo programskega jezika python, za grafični vmesnik pa sem uporabil knjižnico Tkinter, ki je preprosta za uporabo in dobro dokumentirana na spletu.

Načrt

Pred izdelavo programa sem si najprej izdelal preprost načrt o izgledu in funkcionalnosti programa, ko bi ta bil dokončan. Naredil sem spisek željenih funkcionalnosti oz. metod ter skico grafičnega vmesnika. Glavne funkcionalnosti oz. elementi, ki sem jih želel v programu, so bili:

- polja za vnos potrebnih podatkov (vir podatkov za treniranje in testiranje, velikost vhodnih slik, velikost serije, število epoh…),
- polje za sestavljanje modela CNN,
- izpisno polje in
- polja za prikazovanje podatkov o treniranju in samem modelu.

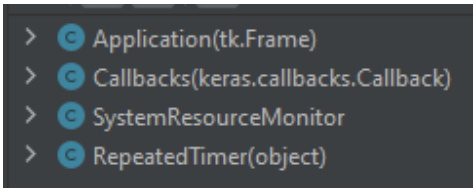
Izgled grafičnega vmesnika sem si skiciral v slikarju, kot je prikazano na sliki spodaj, pri čemer zeleno polje predstavlja polje za sestavljanje modela, rdeče polje vire podatkov za treniranje in testiranje, vijolično polje polja za vnos podatkov o samem modelu, modro polje izpisno polje in rumeno polje pa za izpis podatkov o treniranju in modelu.



Slika 12: Postavitev elementov v grafičnem vmesniku

Izdelava programa

Za izdelavo programa sem uporabil pythonovo knjižnico Tkinter, s katero pa v preteklosti še nisem veliko delal. Zato sem si pri programiranju pomagal z dokumentacijo knjižnice, ki je na voljo na spletu in sem jo tudi navedel med viri. Odločil sem se za objektni pristop izdelave programa in si s tem olajšal izdelavo in organizacijo programa. Aplikacijo sem razdelil v štiri glavne razrede: razred Application, kjer se nahaja grafični vmesnik in večino funkcionalnosti programa, razred Callbacks, ki je del učenja modela CNN, razred SystemResourceMonitor, ki služi za nadzor porabe računalniških resursov kot je poraba CPU, delovnega pomnilnika, ter razred RepeatedTimer, ki pa služi kot časovnik za nadzor porabe računalniških resursov.



Slika 13: Razredi programa za učenje

Prvi problem, ki sem se ga lotil je bila izdelava grafičnega vmesnika. Gradnike (ang. *widgets*) sem v aplikaciji razporedil s pomočjo okvirjev (ang. *frame*) in mreže (ang. *grid*). Gradnike aplikacije sem razporedil v dva glavna okvirja, v okvir za sestavljanje modela

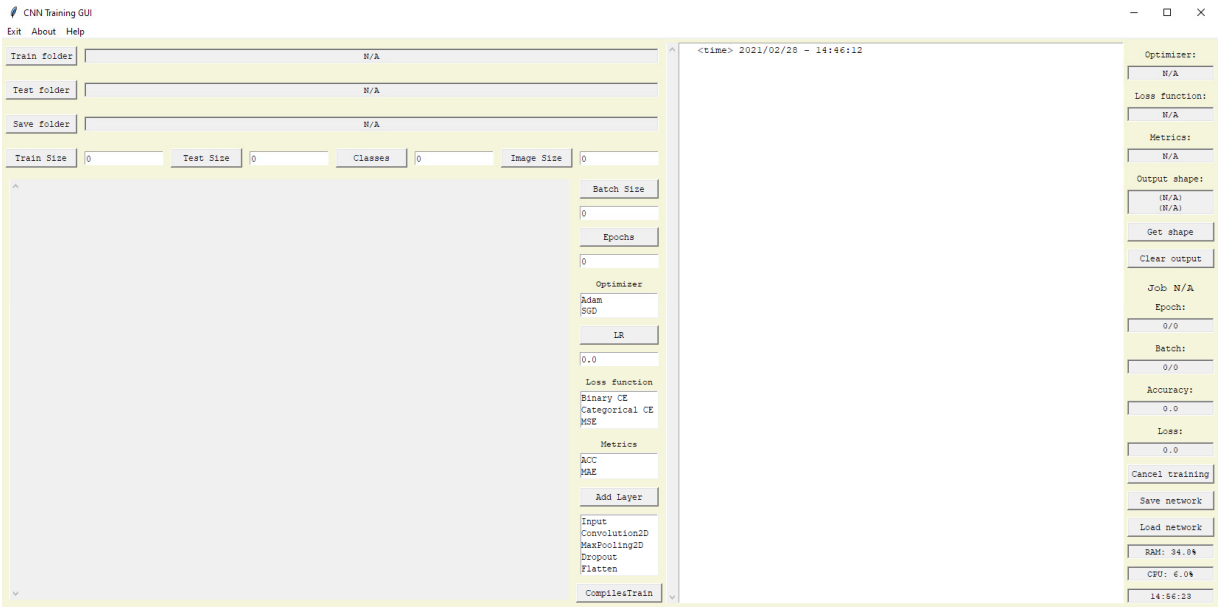
na levi (vijolični, rdeči in zeleni štirikotnik na sliki 12) in v okvir za treniranje modela na desni (moder in rumen štirikotnik na sliki 12). Znotraj teh dveh glavnih okvirjev sem zaradi lažjega razporejanje ustvaril še več okvirjev, ter jih, vključno z gradniki v okvirjih, razporedil s pomočjo mreže. Velik izziv na katerega pa sem naletel pri izdelavi grafičnega vmesnika pa je bila izdelava pomičnega okvirja, v katerega bi lahko zlagal gradnike za sestavo modela CNN. Imel sem nekaj težav pri njegovi izdelavi, a s pomočjo brskalnika in dokumentacije sem kmalu ugotovil, kako ga ustvariti.

Zatem sem se lotil izdelave funkcionalnosti programa. Odločil sem se, da bo program vsak nastali sloj shranil v tabelo. Vsak sloj znotraj tabele je vseboval tudi podatke o sebi, kot so vrsta aktivacijske funkcije, število nevronov na sloju itd. Ostale podatke o modelu, kot je število podatkov za učenje, število dob učenja itd., sem shranil v posamezne spremenljivke, ki pa so sem jih prikazal v zato namenjenih poljih v grafičnem vmesniku.

Programu sem dodal tudi mehanizem za preverjanje veljavnosti modela, ki je preveril vsak sloj za morebitne pomanjkljivosti ali napake. V primeru, da mehanizem v modelu ni odkril nobenih napak, je program poskušal dejansko izdelati model in pričeti proces učenja.

Podajanja informacij uporabniku o procesu učenja modela CNN pa sem se lotil z uporabo po meri narejenih povratnih klicev (ang. *callbacks*). Programu sem dodal funkcionalnost, da ta uporabnika stalno obvešča o procesu učenja, torej kolikšen del učenje je že bil izveden in kolikšno natančnost model ima ob določenem času učenja itd.

Za nameček pa sem v program dodal še grafični prikaz porabe resursov na sistemu uporabnika, ki, prikazuje uporabo procesorskega časa v procentih, porabo delovnega pomnilnika in pa trenuten čas.



Slika 14: Končen izgled programa za treniranje modelov CNN

Backend

Za izdelavo modela in proces učenja sem uporabil pythonovo knjižnico Keras, ki omogoča enostavno izdelavo modelov strojnega učenja. Keras je zelo preprosta, odlično dokumentirana in zelo popularna knjižnica v svetu strojnega učenja. Izdelava modelov z njeno pomočjo je zelo poenostavljena, zato je bila pretvorba iz modela, ki ga uporabnik izdela v grafičnem vmesniku, v dejanski model strojnega učenja ravno tako enostavna.

Za zaledni sistem učenja modela sem sprva uporabil PlaidML. Gre za malce manj znan zaledni sistem, kot je npr. Tensorflow GPU. A za uporabo PlaidML sem se najprej odločil predvsem zaradi tega, ker bazira na OpenGL tehnologiji, ta pa je podprta na vseh modernih grafičnih karticah. Grafične kartice namreč lahko proces učenja modela izjemno pohitrijo. Tensorflow GPU je po drugi strani ob času pisanja te raziskovalne naloge omogočal pohitritev procesa učenja modela le s pomočjo tehnologije CUDA, ki pa jo podpirajo le grafične kartice proizvajalca Nvidia. Ravno zato, ker si ob začetku raziskovalne naloge še nisem lastil tovrstne grafične kartice, sem se sprva odločil za PlaidML. Šele kasneje sem PlaidML nadomestil s Tensorflow GPU, saj sem v tem času zamenjal strojno opremo mojega osebnega računalnika. Za zamenjavo zalednega sistema pa sem se odločil zato, ker je Tensorflow GPU na moji zdajšnji strojni opremi bolj zmogljiv zaledni sistem kot pa PlaidML.

Proces učenja modela CNN

Ko sem imel programa za treniranje in testiranje modela CNN izdelana, sem ju uporabil pri iskanju in preizkušanju modelov CNN, da bi našel takega, ki bil zmožen čim bolje razlikovati med različnimi simboli.

Priprava podatkov za treniranje

Največ dela pri izdelovanju modela strojnega učenja je običajno posvečenega obdelavi, zbiranju in organizaciji podatkov, na podlagi katerih se bo model učil. Tudi sam sem zato potreboval ogromno količino podatkov oz. v mojem primeru slik, ki bi jih posredoval modelu. Odločil sem se da uporabim set 62 različnih na roko napisanih simbolov, kar vključuje celotno angleško abecedo malih in velikih črk ter števk. Uporabil sem *NIST Handprinted Forms and Characters* set podatkov, ki je dostopen na naslednji povezavi: https://catalog.data.gov/dataset/nist-handprinted-forms-and-characters-nist-special-database-19 (21. 4. 2020). Set podatkov je zelo obsežen, kar je pomembno, saj je uspešnost modela med drugimi odvisna tudi od količine podatkov, ki so na voljo za učenje. Več podatkov kot imamo, bolj uspešen bo naš model. Podatki so delno že bili razvrščeni v 62 različnih direktorijev, vsak direktorij za en simbol, a niso še bili razvrščeni v učno podatkovno množico (ang. *training data set*) in podatkovno množico za testiranje (ang. *testing data set*), zato sem le-te najprej rabil razvrstiti v dva direktorija, za učenje in testiranje. To sem storil s pomočjo manjše python skripte, da podatkov nisem potreboval razvrščati ročno.

```
<model> _________________________________________________________________
<model> Layer (type)                 Output Shape              Param #
<model> =================================================================
<model> conv2d_1 (Conv2D)            (None, 126, 126, 32)      896
<model> _________________________________________________________________
<model> conv2d_2 (Conv2D)            (None, 124, 124, 32)      9248
<model> _________________________________________________________________
<model> max_pooling2d_1 (MaxPooling2 (None, 62, 62, 32)        0
<model> _________________________________________________________________
<model> dropout_1 (Dropout)          (None, 62, 62, 32)        0
<model> _________________________________________________________________
<model> conv2d_3 (Conv2D)            (None, 60, 60, 64)        18496
<model> _________________________________________________________________
<model> conv2d_4 (Conv2D)            (None, 58, 58, 64)        36928
<model> _________________________________________________________________
<model> conv2d_5 (Conv2D)            (None, 54, 54, 64)        102464
<model> _________________________________________________________________
<model> max_pooling2d_2 (MaxPooling2 (None, 27, 27, 64)        0
<model> _________________________________________________________________
<model> dropout_2 (Dropout)          (None, 27, 27, 64)        0
<model> _________________________________________________________________
<model> conv2d_6 (Conv2D)            (None, 25, 25, 128)       73856
<model> _________________________________________________________________
<model> conv2d_7 (Conv2D)            (None, 23, 23, 128)       147584
<model> _________________________________________________________________
<model> conv2d_8 (Conv2D)            (None, 19, 19, 128)       409728
<model> _________________________________________________________________
<model> max_pooling2d_3 (MaxPooling2 (None, 6, 6, 128)         0
<model> _________________________________________________________________
<model> dropout_3 (Dropout)          (None, 6, 6, 128)         0
<model> _________________________________________________________________
<model> flatten_1 (Flatten)          (None, 4608)              0
<model> _________________________________________________________________
<model> dense_1 (Dense)              (None, 4096)              18878464
<model> _________________________________________________________________
<model> dense_2 (Dense)              (None, 1024)              4195328
<model> _________________________________________________________________
<model> dense_3 (Dense)              (None, 62)                63550
<model> =================================================================
<model> Total params: 23,936,542
```

Slika 15: Zgradba mojega modela CNN (slika mojega programa za učenje)

Zgradba modela CNN

Zatem je na vrsto prišla odločitev, kakšna naj bo zgradba modela, da bo ta lahko čim bolj natančno predvidel, kateri simbol se nahaja na sliki. Odločil sem se za preprost model CNN, kjer sem uporabil le sloje, ki so v knjižnici Keras poimenovani kot MaxPooling2D, Convolution2D, Dropout, Flatten in Dense. S preizkušanjem različnih modelov sem pričel, ko sem imel izdelan načrt in pripravljene potrebne podatke.

Najprej sem v programu izbral pot do podatkov, torej slik simbolov, ter direktorij na računalniku, kamor se bo model ob končanem učenju shranil. Velikost učne množice in množice za testiranje sem nastavil na privzeto vrednost, torej na maksimum. To pomeni da je model v procesu učenja uporabil za učenje vse slike, ki so mu bile na voljo.

```
<model> Epoch 1
<model> Loss:     0.5293936010988224
<model> Val loss: 0.32159877106563056
<model> Acc:      0.8287489791261836
<model> Val acc:  0.882971082066042
<model>
<model> Epoch 2
<model> Loss:     0.3417025610289494
<model> Val loss: 0.2790251720915692
<model> Acc:      0.8768395649982812
<model> Val acc:  0.8969902565168392
<model>
<model> Epoch 3
<model> Loss:     0.3143529111552317
<model> Val loss: 0.2556800593022377
<model> Acc:      0.8849402214266964
<model> Val acc:  0.9058906995857593
<model>
<model> Epoch 4
<model> Loss:     0.29703289864170884
<model> Val loss: 0.24948897814250012
<model> Acc:      0.8899595335586452
<model> Val acc:  0.9069845102302023
<model>
<model> Epoch 5
<model> Loss:     0.2844605685007286
<model> Val loss: 0.2394604228939025
<model> Acc:      0.8940258272871219
<model> Val acc:  0.9104524444405648
<model> Training finished!
```

Slika 16: Izpis informacij o procesu učenja

V program sem na podlagi zgradbe direktorija, kjer se nahajajo podatki, vgradil tudi funkcionalnost samodejne določitve števila razredov (ang. *classes*), torej števila različnih simbolov. Velikost slik, ki jih posredujem modelu, sem nastavil na 128 x 128, saj je to tudi originalna velikost slik v uporabljeni učni množici. Na ta način nisem izgubljal podrobnosti na slikah. Slik pred učenjem nisem dodatno obdeloval, saj so že bile črno bele in zatorej primerne za takojšnjo uporabo. Velikost serije (ang. *batch size*) sem nastavil na 64, število dob učenja (ang. *epochs*) pa na 5. Optimizacijski algoritem, ki sem ga izbral v mojem modelu, je Adam, funkcija izgube kategorična navzkrižna entropija (ang. *categorical cross-entropy*), metrika za določitev uspešnosti modela pa natančnost (ang. *accuracy*). *Learning rate* sem nastavil na 0,0005. Zatem sem

sestavil model, katerega zgradba in lastnosti so prikazane na sliki 16, in s pritiskom na gumb zagnal proces učenja modela.

Proces učenja je trajal približno 10 ur, po končanem učenju pa je model dosegel natančnost 91 %, kar pa je po mojem mnenju zadovoljiv rezultat, sploh če upoštevamo, da je bil model naučen razlikovati med 62 različnimi, ročno narisanimi simboli, ki so precej zahtevni za prepoznavo.

Model se je tako po končanem procesu učenja shranil v vnaprej določen direktorij na mojem računalniku in bil pripravljen za nadaljnjo uporabo.

Program za testiranje modela CNN

Drugi program, ki sem ga izdelal, je program za testiranje modela CNN. Ta je bil sprogramiran s pomočjo enakih orodij, programskega jezika in knjižnic, kot program za izdelavo modela CNN. Zato postopka izdelave ne bom podrobneje opisoval. Glavna naloga tega programa je torej testiranje izbranega modela CNN.

Pri izdelavi tega programa sem si pomagal z že obstoječim projektom na temo strojnega učenja, ki sem ga že enkrat prej izdelal sam, ko sem s strojnim učenjem šele spoznaval. Program za testiranje je veliko bolj preprost. V njegov uporabniški vmesnik sem dodal dve polji za izbiranje modela in slike, polje za izpis rezultatov testiranja, prikazovalnik izbrane slike in na desni še preprostega risarja, s katerim sem lahko kar znotraj programa narisal črke oz. številke, ki jih je nato izbrani model tudi uporabil.

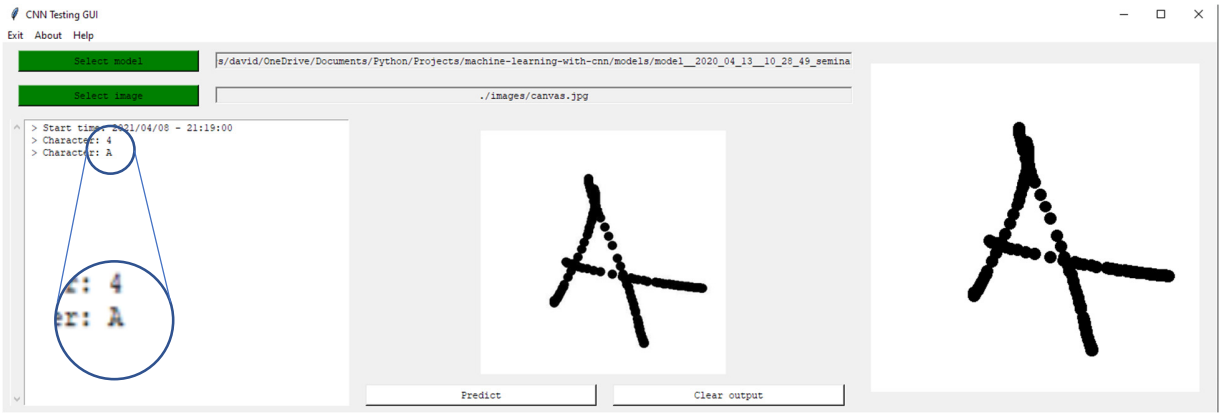
Program je izdelan tako, da samodejno prepozna lastnosti modela in da uporabniku ni potrebno vnašati podatkov, kot so velikost slike, barvna globina itd. Testiranje modela s tem programom je zato bilo zelo enostavno, poleg tega pa za njegovo uporabo, tako kot pri prejšnjem, ni potrebno znanje programiranja.

Slika 17: Izgled programa za testiranje modelov CNN

Testiranje modela CNN

Ko sem torej izdelal model CNN, sem le-tega vstavil v program za testiranje. V program sem nato vstavil še sliko, za katero sem želel, da model ugotovi, kateri simbol je na njej. Program je mišljen le kot grafični vmesnik za napovedi modelov, ki razlikujejo med 62 različnimi simboli. Zato tega programa brez popravkoev ne bi mogli uporabiti za preizkušanje modela, ki bi poleg teh 62 simbolov moral prepoznavati še kaj drugega.

S pritiskom na gumb *Predict* je model tako ustvaril napoved o tem, kateri simbol se nahaja na sliki.



Slika 18: Primer napovedi izdelanega modela z uporabo risarja na desni

Zaključek

Ko je bil model izdelan, sem ga še testiral in ocenil, kako natančen in uporaben je.

Uspešnost modela CNN

Ko se je proces učenja modela končal, je program za izdelavo modela CNN sporočil, da je izdelani model dosegel natančnost 91%. Nad rezultatom sem bil precej navdušen, saj je bil model strojnega učenja, ki sem ga izdelal v tej raziskovalni nalogi, eden izmed bolj natančnih v primerjavi z mojimi prejšnjimi modeli. Poleg tega so bili podatki, ki sem jih uporabil za učenje mojega modela CNN, precej zahtevni za prepoznavo. Hkrati pa je moral razlikovati med kar 62 različnimi simboli. Zato sem kot začetnik na področju strojnega učenja z natančnostjo modela zelo zadovoljen.

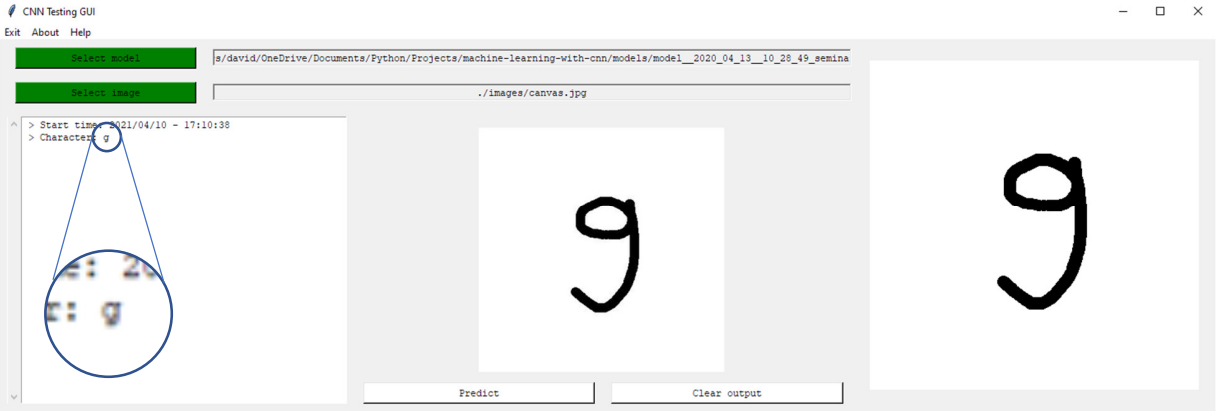
O natančnosti in uporabnosti modela sem se želel prepričati tudi sam, zato sem svoj model preizkusil s pomočjo programa za testiranje na lastnoročno narisanih simbolih abecede in številkah.

Ko sem na modelu preizkusil več simbolov, sem ugotovil, da je večinoma pravilno ugotovil, kateri simbol je na sliki, čeprav so bili simboli mnogokrat izmaličeno narisani. Tudi ko se je poleg simbola nahajala čačka je model večinoma pravilno ugotovil, kateri simbol se nahaja na sliki.

Slika 19: Pravilna ugotovitev modela za črko M kljub čačkam na sliki

Vseeno pa izdelani model ni bil popoln. Ugotovil sem da je model imel težave predvsem v razlikovanju med malo in veliko črko o, ter številko 0. Vsi ti simboli so si med sabo namreč

zelo podobno in pogosto ljudje iz konteksta ugotovimo, ali je simbol v nekem besedilu črka ali številka. Izdelani model pa je opazoval le en posamičen simbol brez konteksta, zato je bilo zanj razlikovanje med črko o in številko 0 skoraj nemogoče. Model je zato skoraj vedno napovedal, da se na sliki nahaja številka 0, pa četudi je bil simbol na sliki dejansko črka o. Model je imel manjše težave tudi pri razlikovanju male črke l, številke 1 in številke 7. Vsi našteti simboli so namreč pogosto zapisani na skoraj enak način, še posebej, če je pisava osebe zelo neberljiva. Model je imel težave tudi pri razlikovanju med črkama g, q in številko 9.



Slika 20: Napačna napoved modela za številko 9

Vse ostale simbole je model uspešno prepoznal in zato menim, da evaluacija modela, ki pravi da je model 91% natančen, drži.

Presenečen sem bil predvsem nad tem, da je model v veliki meri uspešno razlikoval celo med malimi in veliki črkami. Težave so se pojavile le pri razlikovanju male črke f od velike črke F in ostalih črkah, katerih velike črke so podobne malim črkam.

Model sem nazadnje testiral še tako, da sem svoje ime razdelil v črke, ki le-tega sestavljajo in z njimi v programu za risanje sestavil svoje ime. Slike črk mojega imena sem nato posredoval modelu in le-ta je pravilno predvidel, katere črke sestavljajo moje ime. Edina napaka, ki pa jo je model storil je bila ta, da je model predvidel, da je srednja črka mojega imena velika črka V in ne mala črka v.

Slika 21: Primeri ugotovitev izdelanega modela

Hipoteze

1. Prvo hipotezo sem potrdil, saj sem uspel izdelati model CNN z natančnostjo 91% brez potrebe po izdelavi zapletenega modela CNN in le z uporabo preprostih slojev kot sta Convolution2D in MaxPooling2D.
2. Drugo hipotezo sem potrdil z izsledki te naloge in s preizkušanjem različnih simbolov, ki sem jih narisal sam.
3. Tretjo hipotezo sem potrdil tako, da sem oba grafična vmesnika za oba programa, tako za učenje kot testiranje modela, uspel izdelati le s pomočjo programskega jezika python in knjižnic Tkinter ter Keras.

Zaključna beseda

Z raziskovalno nalogo sem zadovoljen, saj sem uspešno izdelal model CNN, ki pa z zadovoljivo natančnostjo razlikuje med 62 različnimi simboli angleške abecede in desetiškimi številkami. Poleg tega sem uspešno izdelal tudi dva grafična vmesnika za izdelovanje in testiranje modelov CNN, s katerima pa sem uspel izdelati svoj model in le-tega tudi testirati. Z raziskovalno nalogo sem dokazal, da strojno učenje celo preprosti napravi kot je računalnik, ki razlikuje le med dvema stanjema, 0 in 1, omogoča opravljanje kompleksnih nalog, kot je prepoznavanje simbolov na slikah. Takšna naloga je za človeka namreč preprosta, a za računalnik precej zahtevna. Z raziskovalno nalogo sem potlej

potrdil tudi, da bo v prihodnosti mogoče skoraj vse dejavnosti, ki jih danes opravlja človek, opravljati tudi s pomočjo računalnika.

V prihodnosti bi želel izdelati še bolj natančen model za prepoznavanje simbolov na slikah, kar pa bi lahko dosegel s pomočjo uporabe dodatnih slojev v modelu CNN, z nadaljnjim treniranjem modela in spreminjanjem parametrov slojev in modela. Poleg tega bi programoma za izdelavo in treniranje modela CNN v prihodnosti želel dodati še več funkcionalnosti in ju tako izboljšati.

Celoten projekt je dostopen preko spletne strani Github, na spodnji povezavi: https://github.com/DavidJerman/machine-learning-with-cnn.

Slovar angleških izrazov

- supervised machine learning – nadzorovano strojno učenje,
- unsupervised machine learning – nenadzorovano strojno učenje,
- semi-supervised machine learning – delno nadzorovano strojno učenje,
- reinforcement machine learning – vzpodbujevalno strojno učenje,
- activation function – aktivacijska funkcija v nevronu,
- propagation function – vhodna funkcija v nevronu,
- input function – vhodna funkcija v nevronu,
- output function – izhodna funkcija v nevronu,
- weight – utež na povezavi med dvema nevronoma,
- input layer – vhodni sloj v umetni nevronski mreži,
- hidden layer – vmesni, skriti sloj v umetni nevronski mreži,
- output layer – izhodni sloj v umetni nevronski mreži,
- widget – gradnik v grafični aplikaciji,
- frame – okvir v grafični aplikaciji, kamor je mogoče zlagati gradnike,
- grid – mreža v grafični aplikaciji, s katero lahko razporejamo gradnike,
- callbacks – povratni klici, metode, ki se kličejo ob določenih dogodkih,
- backend – zaledje programa,
- training data set – učna množica podatkov za učenje (modela),
- testing data set – množica podatkov za testiranje (modela),
- classes – razredi,
- batch size – velikost serije,
- epochs – dobe, epohe,
- categorical cross-entropy – kategorična navzkrižna entropija,
- accuracy – točnost,
- learning rate – hitrost učenja (modela strojnega učenja),
- forward propagation – postopek pošiljanja podatkov »naprej« po umetni nevronski mreži,
- output error – napaka napovedi modela, ki se izračuna v procesu učenja modela,

- backpropagation – metoda vzvratnega razširjanja,
- loss function – funkcija izgube,
- computer vision – veja strojnega učenja, ki se ukvarja z idejo, da računalniki lahko vizualno razumejo svet okoli sebe,
- deep learning – globoko učenje, ena izmed vej strojnega učenja,
- pooling (layer) – združevalni sloj, ki skrbi za dimenzionalno redukcijo,
- flatten (layer) – sploščitveni sloj, ki skrbi za pretvorbo 2D matrike v enodimenzionalni vektor,
- dense (layer) – običajni sloj, ki igra vlogo nevronov v umetni nevronski mreži,
- kernel – filter,
- convolution – konvolucija,
- feature map – matrika lastnosti oz. nabor značilk,
- dimensionality reduction – dimenzionalna redukcija,
- fully-connected layer – polno povezani sloj.

Viri

Nils J. Nilsson. 2010. *The Quest for Artificial Intelligence: A History of Ideas and Achievements.* New York: Cambridge University Press.

Tariq Rashid. 2016. *Make Your Own Neural Network: A gentle journey through the mathematics of neural networks, and marking your own using the Python computer language.* Scotts Valley: CreateSpace Independent Publishing Platform.

C.-C. Jay Kuo. 2016. *Understanding Convolutional Neural Networks with A Mathematical Model.* Cornell University. Dostopno preko: https://arxiv.org/abs/1609.04112 (26. 3. 2020).

Max Tegmark. 2017. *Life 3.0: Being Human in the Age of Artificial Intelligence.* New York: Vintage.

Wikipedia. 2020. Wikipedia. *Machine learning.* Dostopno preko: https://en.wikipedia.org/wiki/Machine_learning (3. 4. 2020).

Python. 2020. Python. *Tkinter – Python interface to Tcl/Tk.* Dostopno preko: https://docs.python.org/3.7/library/tkinter.html (4. 4. 2020).

Tutorialspoint. 2020. Tutorialspoint. *Python – GUI Programming (Tkinter).* Dostopno preko: https://www.tutorialspoint.com/python/python_gui_programming.htm (4. 4. 2020).

Jason Brownlee. 2019. Machine Learning Mastery. *A Tour of Machine Learning Algorithms.* Dostopno preko: https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/ (5. 4. 2020).

Luke Dormehl. 2019. Digital Trends. *What is an artificial neural network? Here's everything you need to know.* Dostopno preko: https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/ (9. 4. 2020).

Poonam Ligade. 2017. Kaggle. *Deep Neural Network Keras way.* Dostopno preko: https://www.kaggle.com/poonaml/deep-neural-network-keras-way (10. 4. 2020).

Amyra Sheldon. 2020. Hackernoon. *A Detailed Premier on Machine Learning Algorithms.* Dostopno preko: https://hackernoon.com/a-detailed-guide-for-machine-learning-algorithms-things-you-must-know-hw97324sa (16. 4. 2020).

Wikipedia. 2020. Wikipedia. *Artificial neural network.* Dostopno preko: https://en.wikipedia.org/wiki/Artificial_neural_network#Components_of_ANNs (19. 4. 2020).

Wikipedia. 2021. Wikipedia. *Strojno učenje.* Dostopno preko: https://sl.wikipedia.org/wiki/Strojno_u%C4%8Denje (7. 4. 2021).

Summit Saha. 2018. Towards data science. *A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way.* Dostopno preko: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (28. 4. 2020).

David Možina. 2021. *Argumentirano strojno učenje z uporabo logistične regresije.* Dostopno preko: https://repozitorij.uni-lj.si/Dokument.php?id=97328&lang=slv (7. 4. 2021).

Dostop do mojega projekta je mogoč na naslednji povezavi: https://github.com/DavidJerman/machine-learning-with-cnn (10. 4. 2021).

Viri slik

Slika 1: Nadzorovano strojno učenje; https://hackernoon.com/a-detailed-guide-for-machine-learning-algorithms-things-you-must-know-hw97324sa (19. 4. 2020).

Slika 2: Nenadzorovano strojno učenje; https://hackernoon.com/a-detailed-guide-for-machine-learning-algorithms-things-you-must-know-hw97324sa (19. 4. 2020).

Slika 3: Vzpodbujevalno strojno učenje: https://hackernoon.com/a-detailed-guide-for-machine-learning-algorithms-things-you-must-know-hw97324sa (19. 4. 2020).

Slika 4: Zgradba umetnega nevrona: http://www.cs.nott.ac.uk/~pszgxk/courses/g5aiai/006neuralnetworks/images/actfn001.jpg (19. 4. 2020).

Slika 5: Zgradba umetne nevronske mreže: https://icdn6.digitaltrends.com/image/digitaltrends/artificial_neural_network_1-791x388.jpg (19. 4. 2020).

Slika 6: Zgradba CNN: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (29. 4. 2020).

Slika 7: Konvolucija: http://people.mines-paristech.fr/fabien.moutarde/ES_MachineLearning/Practical_deepLearning-convNets/convnet-notebook.html (29. 4. 2020).

Slika 8: Ekstrakcija pomembnih lastnosti slike s pomočjo konvolucije: http://people.mines-paristech.fr/fabien.moutarde/ES_MachineLearning/Practical_deepLearning-convNets/convnet-notebook.html (29. 4. 2020).

Slika 9: Max pooling in average pooling: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (29. 4. 2020).

Slika 10: Zgradba CNN (LeNet by Yann LeCun): https://medium.com/datadriveninvestor/five-powerful-cnn-architectures-b939c9ddd57b (29. 4. 2020).