**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKUFAKULTET ELEKTROTEHNIKE, RAČUNARSTVA** IINFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

**Sveučilišni studij**

Primjena neuronskih mreža u klasificiranju LEGO kocaka

**Diplomski rad**

**Anto Tufeković**

**Osijek, 2021.**

SADRŽAJ

[1. UVOD 1](#_Toc79664461)

[1.1. Zadatak završnog rada 1](#_Toc79664462)

[2. STROJNO UČENJE 2](#_Toc79664463)

[2.1. Nadzirano učenje 2](#_Toc79664464)

[2.2. Nenadzirano učenje 3](#_Toc79664465)

[2.3. Nadzirano učenje – Problem klasifikacije 4](#_Toc79664466)

[2.4. Neuronske mreže 4](#_Toc79664467)

[2.4.1. Aktivacijske funkcije 6](#_Toc79664468)

[2.4.2. Treniranje neuronskih mreža 8](#_Toc79664469)

[2.5. Proširenje skupa podataka 10](#_Toc79664470)

[2.6. Skupovi podataka i umjetni podatci 13](#_Toc79664471)

[2.7. Python, Keras, PlaidML 16](#_Toc79664472)

[2.8. [[Transfer learning ovdje ili u 4. poglavlju?]] 17](#_Toc79664473)

[3. REZULTATI MODELA, USPOREDBE 18](#_Toc79664474)

[3.1. Rezultati nad sintetičkim podatcima 18](#_Toc79664475)

[3.2. Problem sintetičkih podataka 23](#_Toc79664476)

[4. [[PYTHON KOD, potrebno?, tu sam kontao transfer learning obraditi]] 25](#_Toc79664477)

[5. ZAKLJUČAK 26](#_Toc79664478)

[PRIZNANJA 27](#_Toc79664479)

[LITERATURA 28](#_Toc79664480)

[SAŽETAK 29](#_Toc79664481)

[ABSTRACT 30](#_Toc79664482)

[PRILOG A: Popis korištenih modela i njihovi prikazi 31](#_Toc79664483)

# UVOD

Često se moraju odrediti točni detalji o nekom objektu ili stvorenju, primjerice psa ili mačke gdje vlasnik se ne zna dovoljno o pasminama da točno utvrdi kojoj pripada. Za taj problem mogu poslužiti umjetne neuronske mreže koje su trenirane nad više tisuća različitih slika svih vrsta pasmina domaćih životinja tako da za neku ulaznu sliku (naprimjer koju vlasnik uslika) preda izlaznu oznaku koja govori o vjerojatnostima kojim pasminama životinja na slici pripada. Isti princip prepoznavanja objekta sa slike se može primijeniti na razne stvari u svijetu koje sadrže uočljive razlike kao što su modeli automobila po markama, obitelji vatrenih oružja, vrsta sorte voća (naprimjer jabuke i kruške), vrste glava šarafa te vrste i modeli igrački, koji su objekt promatranja u ovom radu.

Jedno od situacija gdje ova tehnologija može pomoći je prilikom dizajniranja npr. mobilne aplikacije za slijepe osobe, osoba sa poteškoćama vida i osjetila dodira (svi formalni lijekovi sadrže ime u Brailleovom pismu) može iskoristiti ovakve tehnologije za npr. prepoznavanje kutije lijekova po njenim karakteristikama (veličina kutije, boja kutije, šare po kutiji, prepoznavanje teksta na kutiji) da odabere točne lijekove u pripadajućem trenutku.[[dali je ovaj paragraf potreban, možda je loš primjer?]]

Cilj ovog diplomskog rada je istražiti izradu umjetne neuronske mreže za prepoznavanje i klasificiranje LEGO kocaka na slikama, na takav način da svakoj ulaznoj slici da izlaznu oznaku koja govori kojoj klasi pripada (klasifikacija sa više klasa, engl. *multiclass classification*). Potrebno je napraviti pregled problema na koje se naiđe tijekom treniranja takve umjetne neuronske mreže, te postaviti ili izvesti rješenja problema.

U drugom poglavlju rada će se objasniti osnove strojnog učenja i umjetnih neuronskih mreža te osnovna obilježja ulaznih podataka, potrebno je razumjeti kakvi su podatci s kojim se bavi ovaj rad za puno razumijevanje procesa treniranja umjetne neuronske mreže.

Treće poglavlje služi kao prikaz rezultata modela dobiveni nad podatcima iz drugog poglavlja, te sadrži paralele kakve rezultate daje taj model naspram drugih modela iz područja prepoznavanja LEGO kocaka ili slika.

Četvrto poglavlje se većinom bavi objašnjenjem korištenog programskog koda te pojašnjenjem korištenja prijenosnog učenja (engl. *transfer learning*) u svrhe ubrzanog treniranja nove umjetne neuronske mreže.

[[nisam siguran dali je potrebno četvrto poglavlje, u tom dijelu sam kontao pojasniti transfer learning]]

## Zadatak završnog rada

Proučiti i opisati problem klasificiranja LEGO kocaka. Predložiti model i dizajnirati sustav za klasificiranje te ga ispitati na odgovarajućem podatkovnom skupu.

# STROJNO UČENJE

Strojno učenje je disciplina proučavanja korištenja neke vrste algoritma koja može sebe modificirati kroz neke određene akcije da daje traženi izlaz na neki dani ulaz. Te algoritme je potrebno trenirati tj. potrebno ih je učiti prije nego se mogu koristiti za svoje svrhe, te po tome ih dijelimo na dvije većinske grupe: nadzirano učenje i nenadzirano učenje.

## Nadzirano učenje

(engl. *Supervised learning*) Algoritmi koji pripadaju nadziranom učenju se moraju učiti nad određenim skupom podataka, svaki komad informacije u podatcima mora imati svoju oznaku na kojoj piše šta je točno. Kad se algoritam trenira, on se uči nad ovim podatcima sa oznakama, te time može namještati svoje unutrašnje parametre na takav način da određeni podatci utječu na određen način na parametre, te algoritam to pamti preko danih oznaka, pa nakon treniranja kada se pokušava validirati ispravnost algoritma daju mu se podatci bez oznaka, te algoritam mora odrediti pripadajuću oznaku za dane ulazne podatke.

Neki od problema koji spadaju u nadzirano učenje su:

* Klasifikacija – za određen broj ulaznih podataka algoritam mora moći odrediti njihove pripadnosti u kategorija, npr. na temelju dane slike životinje pripadno trenirani algoritam mora moći odrediti kojoj klasi životinja pripada, tipa ako je dana slika psa algoritam određuje koja je sorta pasmine i slično. Ovaj rad za Lego kocke je tipa klasifikacije, za određene slike Lego kocaka određuje se kojem točno modelu pripadaju
* Regresija – na temelju ulaznih podataka pokušava se dobiti pripadajući realni broj koji opisuje najbolje ulazne podatke (za razliku od klasifikacije, regresija daje realan broj dok klasifikacija daje jedan od mogućih izlaznih oznaka), npr. za dane podatke o rabljenom automobilu kao što su kilometraža, marka, model, postotak štete na autu i slično, algoritam može nam dati procjenu vrijednosti automobila
* Prijevod jezika – algoritam prima riječi ili tekst, te njihove prijevode za učenje, te kad se nakon treniranja da neki novi tekst ono mora moći odrediti prijevod danog teksta
* Prepoznavanje grešaka – algoritmu se daju podatci normalnog ponašanja za treniranje, te pri validaciji nad abnormalnim podatcima algoritam mora moći prepoznati visoko odstupanje. Npr. algoritmu se daju slike ispravnih mehaničkih komponenti za trening, te na novim slikama mora uočiti razlike kao što su pukotine ili iskrivljenje materijala[<https://global.canon/en/technology/crack2019.html>]
* Uklanjanje smetnji – algoritmu se daju podatci koji ga upućuju na ciljane smetnje te trenira se tako da izmjenjuje podatke dok se smetnje ne uklone. Npr. algoritam se uči na slikama sa pomućenjem, npr. od kamera koje su se kretale pri snimanju, te trenira se tako da smanji utjecaj pomućenja mijenjanjem slike

## Nenadzirano učenje

(engl. *Unsupervised learning*) Algoritmi za nenadzirano učenje se treniraju na drugačiji način naspram algoritama za nadzirano. Oni za trening primaju podatke bez oznaka, te moraju onda pronaći neku vrstu strukture u danim podatcima, kao što su grozdovi ili skupovi podataka ovisno o informacijama u podatcima, te za validaciju onda moraju točno svrstati nove podatke u dane skupove, iako točnost je ovisna o ljudskoj osobi koja procjenjuje kvalitetu kreiranih grupa.

Glavna korist nenadziranog učenja nad skupovima podatka je procjena gustoće (engl. *density estimation*), algoritam prema podatcima kreira polja koja sadrže neki postotak vjerojatnosti da podatak pripada tom skupu, što je bliže „centru“ polja to je veća vjerojatnost da pripada tom skupu.

Neki od problema koji spadaju u nenadzirano učenje su:

* Grupiranje – algoritmu se daju podatci sa određenim informacijama, algoritam pokušava te podatke grupirati u svoje grupe ovisno o njihovim informacijama, postoji više metoda za grupiranje
* K-means – metoda grupiranja gdje algoritam koristi informacije u podatcima da odredi skupova podataka, isprva odabere nasumičnih podataka te onda iterativno traži prosječnu udaljenost prema svim podatcima koji se smatraju pod tim skupom, nakon toga ponovno izračuna središnje točke svih skupova i ponavlja algoritam dok se ne konvergira u mirno stanje (konačno konvergirano stanje ne mora biti optimalno stanje)

## Nadzirano učenje – Problem klasifikacije

Klasifikacija je problem iz područja statistike, u problemu se pokušava doći do strukturiranog rješenja gdje nekoj promatranoj jedinici damo jednu oznaku iz skupa oznaka. Jedan jednostavan primjer bi bila klasifikacija elektronske pošte u ulazni sandučić ili u neželjenu poštu (engl. *spam*).Takvi ulazi moraju u sebi sadržavati neku informaciju (engl. *feature*) koja algoritmu pomaže odrediti kojoj od klasa ulazni podatak pripada, takve informacije mogu biti same po sebi kategorične (npr. osoba može imati krv tipa „A“, „AB“, „A+“, itd.), mogu biti bazirane na brojevima (npr. slike imaju piksele koje sadrže tri kanala za boje, najčešće sa vrijednostima od 0 do 255) ili jednostavno uspoređivati dolazeće podatke sa prijašnje treniranim podatcima tako da uspoređuje udaljenost ili sličnost novih podataka od treniranih.

Algoritam koji obavlja klasifikaciju se naziva klasifikator, ako postoje samo dvije klase za klasificirati onda se metoda zove „binarna klasifikacija“, ako imamo više klasa metoda se zove „klasifikacija s više klasa“ a ako imamo metodu gdje neki izlaz može imati više klasa pridruženo onda se radi o „klasifikaciji s više oznaka“.

## Umjetne neuronske mreže

Umjetne neuronske mreže (engl. *Artificial neural networks – ANNs*) su algoritmi koji se baziraju na neuronskim mrežama životinja iz pravog svijeta, neuronska mreža se sastoji od kolekcije spojenih jedinica koje se nazivaju neuroni. Spoj među neuronima je sličan spoju kojeg prave pravi živci u obliku sinapsa, samo za olakšano i sigurnije programiranje tih spojeva modelira se sve u obliku slojeva neurona, slojevi su međusobno spojeni i tako se signal na ulazu propagira prema izlazu, kao što je prikazano na slici 2.1.



**Slika 2.1.** Uzor neurona je prava biološka živčana stanica [[izvor](https://commons.wikimedia.org/wiki/File:Neuron3.png) slike]

Na slici 2.2. se vidi da svaki neuron po uzoru akson – sinapsa – dendrit (izlaz – prijenos – ulaz) ima svoje ulaze () na dendritima, neka funkcija stanice u tijelu, te onda preko aksona ima izlaz() kojim se dalje propagira do kraja sustava. Prethodno spomenuta funkcija stanice neurona određuje kakav će izlaz biti na svakom izlazu ovisno o ulazu, funkcija sadrži u sebi težine koje se pri treningu mijenjaju ovisno o stimulusu te onda odlazi u aktivacijsku funkciju. Najjednostavnije objašnjenje je da svaki izlaz neurona je realna funkcija, te težina pojačava ili oslabljuje „signal“ na danom izlazu.



**Slika 2.2.** Matematički model neurona[[izvor slike](https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc)]

Osnovni matematički model neurona je da svaki ulaz ima svoju težinu [[izvor za ovu sekciju](http://didattica.cs.unicam.it/lib/exe/fetch.php?media=didattica:magistrale:kebi:ay_1718:ke-11_neural_networks.pdf)], težine se postavljaju tijekom treninga ovisno o važnosti ulaza. Na kraju neuron izvršava neku funkciju (često zvana aktivacijska funkcija) nad sumom svih ulaza množeni svojim težinama te to predstavlja kao izlaz svim sljedećim spojenim neuronima. [[Ovaj dio bilo bi potrebno uklopiti sa ostatkom teksta]]



**Slika 2.3.** Primjer neuronske mreže sa jednim skrivenim slojem

Kao što je na početku poglavlja spomenuto, neuroni su često poslagani u slojevima zbog jednostavnosti kreiranja modela, to ne znači da signal putuje jednom kroz model i završava odmah na izlazu, nego u modelu se mogu nalaziti razne funkcionalnosti između slojeva koje pomažu modelu postići veći uspjeh te glavni cilj tih slojeva je izbjegavanje mrtvih neurona. [[Prvo moramo pregledati kakve aktivacijske funkcije postoje i kako mogu pomoći. OVO možda pomaknuti, pokazati treniranje prvo i zapisati tekst za taj dio pa micati okolo stvari?]]

### Aktivacijske funkcije

Aktivacijske funkcije korištene u umjetnim neuronima djeluju nad sumom ulaza množeni sa njihovim pripadajućim težinama [[izvor za ovu sekciju](https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/)]. Te funkcije daju drugačije izlaze ovisno o treniranim parametrima težine u neuronima.

Najčešće korišteni u praksi su:

* ReLu aktivacija (ispravljena linearna jedinica, engl. *rectified linear unit*) (raspon ([0, ∞)) i njena derivacija:

|  |  |
| --- | --- |
|  | (2-1) |

* Sigmoid(logistička) funkcija (raspon (0,1)) i njena derivacija:

|  |  |
| --- | --- |
|  | (2-2) |

* Tanh aktivacijska funkcija (raspon (-1, 1)) i njena derivacija:

|  |  |
| --- | --- |
|  | (2-3) |



a) b) c)

**Slika 2.4.** Prikaz aktivacijskih funkcija: a) ReLu (2-1), b) Sigmoid (2-2), c) Tanh (2-3)

Kod ove tri aktivacijske funkcije važno je napomenuti da se može dogoditi problem sa nestajućim gradijentom prilikom treniranja (više objašnjeno u sljedećem podpoglavlju). Glavni princip problema je da kod Sigmoid i Tanh funkcija (slika 2.4. b) i c)) imamo lokalni maksimum i minimum (Sigmoid ima raspon (0,1) dok Tanh ima (-1,1)), ako se preko SGD (ili nekog drugog algoritma optimizacije) i povratnog razmnožavanja naiđu brojevi veći od maksimuma i minimuma, oni se prevedu natrag direktno u maksimum/minimum, te time se onemogućava daljnje treniranje neurona. ReLu ima sličan problem ali samo kod minimuma (slika 2.4. a)), zato je njegov raspon [0, ∞) te to uvelike pomaže kod treniranja dubokih mreža jer mreže koje se zasnivaju na Sigmoid ili Tanh funkcijama često naiđu na problem nestajućeg gradijenta, jer gradijent se kreće u nekom smjeru pa ako naiđe na minimum ili maksimum pojavljuje se onda problem. ReLu je otporan na maksimume te nema takve probleme.

Ako se trenira mreža sa ReLu i naiđe se na problem nestajućeg gradijenta, moguće je koristiti njegovu alternativu, parametarski ReLu (engl. *parametric ReLu*, slika 2.5.).



**Slika 2.5.** Prikaz aktivacijske funkcije parametarski ReLu

Karakteristika ove aktivacijske funkcije je da više nema raspon od 0 do beskonačno, nego mu je modificiran negativan dio koristeći parametar (često je jednak 0.01 pod nazivom „propusni“ tj. engl. *leaky ReLu*):

|  |  |
| --- | --- |
|  | (2-4) |

Na ovaj način se može izbjeći problem nestajućeg gradijenta jer više ne može zapeti na nuli, nego se može kretati slobodno pa postoji šansa da se težina oživi natrag iznad nule. Neuron kojemu su sve težine zapele na negativnoj strani se naziva mrtav neuron, te propusni ReLu služi kao mogućnost oživljavanja takvih neurona.

### Treniranje neuronskih mreža

Treniranje umjetnih neuronskih mreža se izvodi preko stohastičkog gradijentnog spusta (engl. *stochastic gradient descend*) iliti SGD, SGD je iterativna metoda za optimizaciju funkcije gubitka treninga, sa dobivenim gradijentom ono može zajedno sa algoritmom povratnog razmnožavanja [[izvor](http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf) 9-48] prolaziti unatrag kroz sve slojeve neuronske mreže, mijenjajući težine na ulazu neurona. Za ovu svrhu je potrebno da je aktivacijska funkcija neurona derivabilna jer algoritam kako prolazi unatrag mora moći proći i kroz aktivacijsku funkciju neurona da može nastaviti.

Ovdje se pojavljuje problem u treniranju neuronskih mreža, naime problemi nestajućeg i eksplodirajućeg gradijenta [[izvor za ovu sekciju](https://books.google.hr/books/about/Deep_Learning.html?id=Np9SDQAAQBAJ&printsec=frontcover&source=kp_read_button&redir_esc=y#v=onepage&q&f=false)]. Kako se trenira vrlo duboka neuronska mreža sa mnogo slojeva, postoji vjerojatnost da će gradijent, kako prilazi početku umjetne neuronske mreže, biti toliko malen da će imati neznatan utjecaj na težine, u najgorem slučaju može se dogoditi da se neuronska mreža kompletno prekine trenirati zbog ovog problema. Primjer događanja bi bio da neka aktivacijska funkcija, primjerice Tanh koja ima gradijente samo u rasponu od 0 do 1, kroz povratno razmnožavanje smanji svoj gradijent na eksponencijalno male brojeve kako prolazi kroz slojeve, time uzrokuje da rani slojevi dobiju vrlo malo ili ništa treninga. Sličan problem se ukazuje kod eksplodirajućeg gradijenta kada derivacije aktivacijskih funkcija primaju vrlo visoke vrijednosti, što uznemiruje ravnotežu treniranja sa divljim mijenjanjem težina.

Postoje varijacije SGD algoritma koje sadrže moguća poboljšanja nad osnovnim SGD algoritmom kao što su SGD sa adaptivnim gradijentom[[izvor](https://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf)] (engl. *AdaGrad*), SGD gdje se stopa učenja mijenja za svaki parametar u mreži[[izvor](http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf)] (engl. *RMSProp – Root Mean Square Propagation*) te Adam algoritam [[izvor](https://arxiv.org/pdf/1412.6980.pdf)] (engl. *Adaptive Moment Estimation*). Adam algoritam je nadogradnja *RMSProp* algoritma na takav način da ukomponira svojstva *AdaGrad* algoritma, umjesto da samo bazira stopu učenja za svaki parametar prema srednjoj vrijednosti za svaki parametar, ono uzima u obzir i adaptivni gradijent *AdaGrad* algoritma u obliku momenta gradijenta. Taj moment pomaže algoritmu ubrzati pronalazak optimalne težine za svaki parametar (ubrzava pronalazak točke konvergencije gradijentnog spusta). Adam se koristi pri treniranju dubljih neuronskih mreža sa više skrivenih slojeva jer sadrži manju vjerojatnost kreiranja mrtvih neurona zbog prethodno spomenutih učinaka.

Ciklus treniranja nad skupom podataka se naziva epoha, svaka epoha se provodi nad ulaznim skupom podataka iz kojeg se odabire mini-šarža (engl. *mini-batch*) podataka, koristeći SGD i povratno razmnožavanje ovaj se postupak iterativno ponavlja dok se ne iskoriste sve mini-šarže u skupu podataka. Na kraju epohe se na validacijskom skupu podataka provjeravaju performanse mreže, ti parametri se onda mogu koristiti u sljedećim epohama za namještanje parametara treninga. Kad završi trenutna epoha pokreće se sljedeća dok ne prođu sve zadane epohe. Algoritam treniranja može rano zaustaviti treniranje ako vidi da se funkcija greške ne mijenja znatno preko vremena, ovo je naznaka da je algoritam treniranja došao u lokalni minimum za moguću grešku, te potrebno je dalje namještati parametre ili ako su zadovoljavajući rezultati spremiti model za daljnje korištenje.

Daljnji problemi pri treniranju se mogu pojaviti u obliku pretreniranja (engl. *overfitting*). Pretreniranje je pojava u kojoj trenirani model postaje previše naviknut na skup podataka na kojem je treniran. To se događa ako je skup podataka za treniranje vrlo malen, ili ako su podatci vrlo slični jedni drugom. Prilikom treniranja mreže potrebno je obratiti pažnju na parametre koje se vraćaju na kraju epohe, koje govore kako model u toj epohi djeluje nad validacijskim skupom. Ako model ima dobre performanse nad trening skupom ali loše performanse nad validacijskim skupom može se reći da pati od pretreniranja nad trening skupom.



**Slika 2.6.** Grafički prikaz za utjecaj parametara treninga na pretreniranje[[izvor slike](https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning)]

Na slici 2.6. se vide primjeri tri modela sa drugačijim parametrima treniranja. Prvi model je oblika polinomna regresija sa jednim stupnjem, zbog toga se ne može točno podatcima namjestiti (engl. *underfit*). Drugi model ima četiri stupnja, što je dostatno da se model namjesti trening podatcima i da liči originalnoj funkciji. Treći model ima previše stupnjeva te pokušava se namjestiti na takav način da kroz sve podatke pokušava proći. Tu se nalazi problem pretreniranja.

Sličan koncept se može dogoditi i kod neuronskih mreža, ako preko SGD i povratnog razmnožavanja konstantno dolaze slični gradijenti, neuroni će biti naučeni na te specifične podatke, te kad dođu neki drugi dovoljno drugačiji podatci, neuroni neće biti u mogućnosti dati kvalitetan odgovor prema sljedećem sloju, što uzrokuje grešku u npr. predviđanjima i pogoršava izlazne parametre tijekom validacije. Otkrivanje podtreniranja i pretreniranja je jednostavno ako imamo parametre sa treniranja, ako model ima dobre performanse prilikom treniranja i dobre performanse pri validaciji, onda se može reći da je model kvalitetan. Ako model ima loše performanse kod validacije onda je moguće pretreniran, dok loše performanse kod treniranja ukazuju da nije dovoljno treniran, tj. podtreniran je i nedostaje više epoha treniranja.

## Proširenje skupa podataka

Za potrebe treniranja neuronskih mreža može se dogoditi situacija da ne postoji dovoljno dostupnih podataka nad kojima se može trenirati, razni su razlozi kao npr. poštivanje privatnosti u medicini (naprimjer rendgenske, mamografske, ultrazvučne slike i slično) zbog čega gubimo pristup vrijednim podatcima za treniranje, a potrebna je velika količina da se može trenirati kvalitetna mreža, pogotovo ako je duboka mreža, što je dublja to je više raznolikosti u trening podatcima potrebna da se kvalitetno trenira bez problema pretreniranja i podtreniranja.

Za tu svrhu postoji proširenje skupa podataka (engl. *Data augmentation*). To je svaka vrsta tehnike proširenja postojećeg skupa podataka tako da se uvede nasumično transformirana kopija nekog originala iz skupa ili se generira sintetički podatak od originalnih[[izvor](https://www.researchgate.net/profile/Agnieszka-Mikolajczyk-3/publication/325920702_Data_augmentation_for_improving_deep_learning_in_image_classification_problem/links/5d5d5569458515210257607c/Data-augmentation-for-improving-deep-learning-in-image-classification-problem.pdf) za ovo cijelo podpoglavlje? sam izvor ima svoje izvore, možda izvučem nešto]. Za kreiranje nasumične kopije originala često se koriste jednostavne funkcije kao što su afine transformacije (rotacija, skaliranje, translacija, refleksija, smicanje), transformacije nad histogramom (npr. ujednačenje), te mijenjanje svojstva slike kao što su kontrast, svjetlina ili transformacije kao povećanje oštrine, pomućenje slike i izjednačavanje svjetline.

Razlog za proširenjem skupa podataka je poboljšanje generalizacijskog svojstva mreže. Ako se kompleksna mreža trenira nad malim brojem podataka za normalan broj epoha, dogoditi će se podtreniranje mreže jer mreža nije dovoljno trenirana da uspješno prepoznaje objekt na slici, ali ako se trenira nad malim brojem podataka za više epoha mreža će početi pamtiti ulazne podatke i dogoditi će se pretreniranje što opet smeta generalizaciji, jer pri treniranju će imati visoku točnost, a pri validiranju nisku točnost.

a) b)

**Slika 2.7.** a) Slika modela 3003 LEGO kocke, b) Slika modela LEGO kocaka proširene raznim tehnikama

a) b)

**Slika 2.8.** a) Uslikana LEGO kocka 3001, b) Uslikana LEGO kocka 3001 proširena raznim tehnikama

Na slici 2.7. a) se vidi sintetički generirana slika, takva LEGO kocka je slikana, tj. „generirana“, iz raznih kutova. Zato što postoji više modela, od kojih neki su nesimetrični (prilog A) za svrhe proširenja skupa podataka se koriste tehnike: refleksija, translacija, rotacija i smicanje. Refleksiju je potrebno raditi samo u smjeru horizontalne osi (*engl. horizontal flip*), jer rotacija će pokriti sve ostale moguće pozicije. Translacija se koristi za svrhu povećanja regularizacije, ako se mreža trenira nad savršeno centriranim podatcima, onda će i očekivati savršeno centrirane podatke tijekom validacije, koristeći translaciju ovaj se učinak smanjuje. Smicanje služi za simuliranje iskrivljenja leće kamere ili neke druge okolnosti koja uzrokuje nepravilnost modela (naprimjer šteta plastike uzrokovana toplinom).

Na slici 2.7. b) se vidi 3 sa 3 mreža generiranih slika. Nakon što su modeli provedeni kroz proširenje dodaje im se nasumična pozadina od par odabranih slika. Pozadina se izrađuje na način da se izreže željena veličina pozadine iz skupa zadanih slika (tipa 200x200 slika se u nasumičnoj poziciji isječe iz 2000x4000 slike). Time smo dobili skup upotrebljivih slika s kojim se treniraju mreže. Pozadine su odabrane nasumično ali da imaju potrebne teksture koje bi se mogle nalaziti i u pravim slikama za buduću validaciju: slike ljudskih ruku, slike tretiranog drveta (radne površine, vrata), slike zamućenih objekata i drugo.

Na slici 2.8. a) se vidi uslikana slika LEGO kocke 3001. Nad uslikanim slikama imamo manje slobode za manipuliranje npr. pozadinama ali možemo i dalje iskoristiti sve ostale afine transformacije bez problema.

Za podatke koji su uvijek centrirani koriste se sljedeće transformacije:

* Nasumično smicanje: raspon [-10°, 10°]
* Nasumično povećanje (engl. *zoom*) slike, raspon [0%, 20%]
* Nasumična translacija horizontalno, raspon [-10%, +10%]
* Nasumična translacija vertikalno, raspon [-10%, +10%]
* Nasumično mijenjanje svjetline, raspon [50%, 100%] originalne svjetlosti
* Nasumična refleksija horizontalno, vertikalno
* Nasumična rotacija, raspon [0°, 90°]

Koristeći refleksiju u oba smjera zajedno sa nasumičnom rotacijom od 90°, pokrivene su sve mogućnosti pozicije nekog LEGO modela.

## Generiranje skupa umjetnih podataka

Skupovi podataka korišteni u ovom radu se sastoje od umjetno generiranih slika LEGO kocaka. Prije treniranja neuronskih mreža nad slikama ako je moguće za što veću točnost je potrebno što više standardizirati ulaznu sliku LEGO kocke. Ako je pozadina neka vrsta reljefa ili jednostavna tekstura, neuronska mreža može sama sebe trenirati da takve značajke ignorira te time se može koncentrirati na detekciju rubova, detekciju oblika, detekciju broja dugmadi na LEGO kockama i drugih značajki potrebno za klasifikaciju. Neuronske mreže su posebne na taj način da im se ne mora govoriti što da traže, nego one će preko trajanja treninga naučiti same kako otkriti željene značajke.

Umjetni podatci imaju široki raspon svrha za koje se mogu koristiti [[izvor](https://research.aimultiple.com/synthetic-data/)] te najčešće pojednostavljuju i pojeftinjuju operacije za koje su vezane, u slučaju ovog rada se ne mora par dana slikati par LEGO kocaka, nego je moguće generirati sve slike u raznim uvjetima, mogu biti sa raznim pozadinama, mogu dijelovi kocaka biti skriveni, svijetlo na kockama se može namještati, slike se mogu generirati oštećene kao simulacija slikanja u pravom svijetu (primjer je generirana slika sa pomućenjem u nekom smjeru, kao da je osoba koja je slikala micala kameru pri slikanju). [[Umetnuti dodatne sitnice oko podataka tijekom generiranja?]]



**Slika 2.10.** Isječak iz Blender prozora za prikaz lokacija objekta, svjetla i kamere

Sintetički podatci za ovaj rad su generirani u Blenderu koristeći skriptu koja uslika zadani model iz raznih kutova, te slike imaju dva svjetlosna izvora u blizu sredine ali su za jednu prostornu jedinicu odmaknutu svaka u svoju stranu (na Slici 2.10. se vidi da su odmaknute od centra 4 prostorne jedinice, u stranu pomaknute za 1 prostornu jedinicu te visoko/duboko postavljeno 6 prostornih jedinica). LEGO modeli se skaliraju tako imaju točan odnos veličine međusobno (npr. vizualno pregledati da su na svim modelima dugmadi iste veličine, da su sami modeli iz istog izvora, te time garantirano iste skale, da su tehničke kocke iste dužine (2M, 3M, itd.)) i uslikane su sa jednakom udaljenošću od kamere. LEGO modeli su nasumično obojani od par odabranih boja, kao što su plava, ružičasta, siva, itd., neuronske mreže moraju biti otporne na različite boje u slikama, iako jedan mogući zaobilazak problema je da se slike prije treniranja pretvore u slike sivih tonova.



**Slika 2.11.** Prikaz načina kretanja kamere u Blenderu

Kamera slika model počevši od skroz gore (slika 2.11.), te u zadanim koracima se kreće sferično oko modela prema dolje. Kako prolazi tako kamera „slika“ model, te generirane slike sprema za daljnje korištenje u treniranju.

Za rad su korišteni modeli koji su ilustrirani u Prilogu A. Cilj je bio koristiti modele koji imaju različite stupnjeve sličnosti, više o tome u podpoglavlju 3.1..

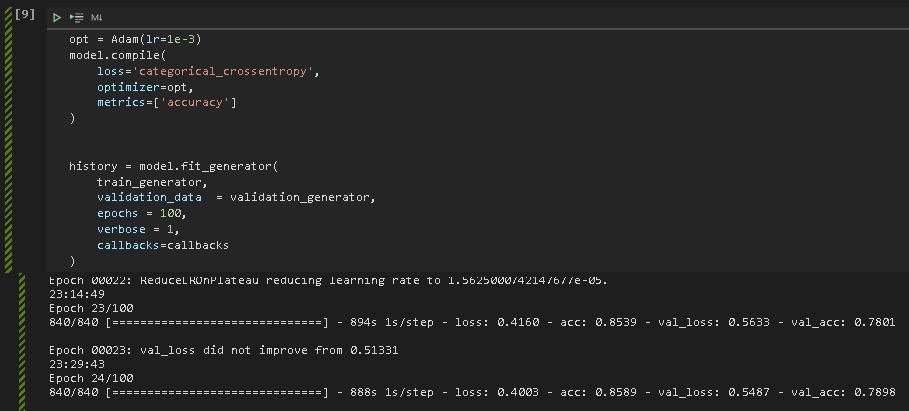
## Python, Keras, PlaidML

Modeli ovoga rada su pisani u *Python* [[izvor](https://www.python.org/)] programskom jeziku koristeći biblioteke *Tensorflow* i *Keras* koje služe za programiranje strojnog učenja [[izvor](https://keras.io/)]. Strojno učenje preko spomenutih biblioteka se može provoditi preko procesora ili grafičke kartice, te korištenje grafičke kartice za treniranje modela je brža opcija naspram procesora (deset puta brže ili čak više). Međutim, treniranje preko grafičke kartice se tradicionalno odvijalo na Nvidia grafičkim karticama jer biblioteke su pisane sa naredbama specifično za te kartice, te za to služi *PlaidML*.

*PlaidML* je biblioteka koja se veže za *Keras* biblioteku na takav način da dopušta raznim grafičkim karticama, kao što su Intel-ove integrirane kartice ili AMD kartice, da preuzmu istu funkciju kao i Nvidia grafičke kartice te ono nam prevodi i kompilira sve njihove naredbe u naredbe koje ostale grafičke kartice razumiju.

*Keras* je biblioteka koja je nastavak na *Tensorflow* biblioteku, ono služi kao posrednik između programera i *Tensorflow* biblioteke te time uvelike olakšava programiranje i namještanje modela te snalaženje u kodu.

Cijelokupno izvođenje *Python* koda se provodi unutar *Jupyter* bilježnica[[izvor](https://jupyter.org/)]. Takve bilježnice se koriste za brzo izvođenje promjena bez potrebe ponovnog paljenja i gašenja *Python* jezgre, te služi nam kao spremnik izlaza *Python* kôda za daljnju analizu (slika 2.13.).



**Slika 2.13.** Prikaz dijela *Jupyter* bilježnice u kojem se izvodi trening mreže

*Blender* je program[[izvor](https://www.blender.org/)] za 3D modeliranje, te sadrži mogućnosti za izradu slika i animacija modela. Koristi se u ovom radu za generiranje slika sintetičkog skupa.

Cijeli projekt za ovaj rad se nalazi na sljedećoj Github poveznici:  
<https://github.com/ATufekovic/LEGO_classification>

Projekt sadrži dva direktorija, *docs* i *py*. Unutar *docs* se nalaze svi materijali vezano za pisani rad i ovaj sam rad. Unutar *py* se nalazi više poddirektorija, svaki poddirektorij sadrži svoju *Jupyter* bilježnicu unutar koje se izvršavao *Python* kôd. Direktoriji sa prefiksom „test“ su direktoriji koji sadrže neuronske mreže nad kojima se treniralo, te sve funkcije vezane za njih. Direktorij „image\_generation“ sadrži *Blender* skriptu i *.blend* datoteku za koju je skripta namijenjena. Datoteka sadrži sve modele i okolinu za slikanje slika, dok skripta služi za automatizaciju generiranja slika. Direktorij „image\_sourcing“ sadrži *Jupyter* bilježnicu za generiranje slika za korištenje u ovom dokumentu.

## [[Transfer learning ovdje ili u 4. poglavlju?]]

# REZULTATI MODELA, USPOREDBE

[[Ovdje bi trebao krenuti opisati *state of the art*, na koji nacin?]]Klasificiranje slika su ljudi izvodili već duže vrijeme, AlexNet[[izvor](https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf)] je jedan primjer neuronske mreže koja je 2012. godine u ImageNet natjecanju široko pojasnog prepoznavanja postigla prvo mjesto, pobijedivši drugo mjesto za više od 10% manje grešaka u 5 najvećih kategorija koje predstavljaju greške[[top-5 error prevesti ljepše?]], najviše jer su koristili treniranje preko grafičkih kartica. Ovo je privuklo globalnu pozornost prema koristi područja umjetnog učenja i umjetne inteligencije te ubrzo je krenulo napredovanje tih tehnologija. Dan danas se rezultati tehnologija nalaze u raznim poljima svakodnevnice, kamere koriste tehnologije prepoznavanja lica i aktivnog stabiliziranja videosnimke, doktori sa modelima mogu lakše uočiti bolesti na raznim vrstama snimaka, od EKG do rendgenskih slika te moguće je lakše prevesti pokrete tijela u pokrete objekata virtualne stvarnosti uz pomoć neuronskih mreža (npr. za koristi u arhitekturi[[izvor](https://www.davidpublisher.com/Public/uploads/Contribute/5d5e0535ad919.pdf)]).

## Rezultati nad sintetičkim podatcima

U poglavlju [[stagod za sintetičke](#_Skupovi_podataka_i)] su pojašnjene koristi i mane sintetički generiranih podataka, za slučaj u radu znamo da LEGO kocke imaju oštro definirane oblike (nikada se dvije kocke istog modela ne razlikuju, osim možda po boji), kocke imaju rubove i dugmadi, ploče imaju glatke površine bez dugmadi, štapići za povezivanje su cilindričnog oblika sa specifičnim reljefima na krajevima, kotači su široki cilindri sa reljefima na kontaktnoj površini, itd.

Na github stranici rada se nalaze *Jupyter* bilježnice sa modelima, te njihove strukture i rezultati.

**Tablica 3.1.** Prikaz rezultata za korištene modele

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | loss | acc | val\_loss | val\_acc |
| test | 2.5988 | 15.80% | 2.4492 | 17.83% |
| test\_complex | 1.0020 | 66.06% | 0.9606 | 63.97% |
| test\_vgg16 | 0.3314 | 88.30% | 0.3880 | 85.97% |
| test\_vgg19 | 0.3692 | 87.06% | 0.4012 | 84.38% |

Nad sintetičkim podatcima jednostavne mreže mogu postići visok rezultat kao što je vidljivo na tablici 3.1.1. Problem koji je spomenut na kraju podglavlja 2.6. je vezan za sličnosti među modelima, uzmimo za primjer model 3001 2x4 kocku i model 3020 2x4 ploču (slike se nalaze u prilogu A, konkretan primjer na slici 3.1.1.). Ako se trenira mreža sa slikama modela od skroz gore ili dolje, model će naučiti da su to značajke ta dva modela, ali kad dođe u pitanje predviđanje modela takve slike, mreža neće moći sa potpunom sigurnošću reći koji je to model osim ako nije slikano dovoljno u stranu, da se vidi debljina modela. Slično se može produžiti na model 3037 2x4 kocke za krov, ako se da slika od dolje to je još jedan sličan model za mrežu, te tu nije teško da se događaju greške, zbog toga nije moguće da mreža postigne neku vrlo visoku točnost. Paralelno tome je odabrano još par modela koji su ili slični ili jedinstveni u izgledu, naprimjer model 2340 kormilo i 4083 ograda, ti modeli su iz skoro svih pogleda relativno jedinstveni, dok postoje dodatni parovi za otežavanje predviđanja mreže kao što su model 2357 1x2x2 rub kocka i model 2420 1x2x2 rub ploča, gdje je jedina razlika u debljini modela.

a) b)

c) d)

**Slika 3.1.** Prikaz modela 3001 i 3020 za usporedbu poteškoće predviđanja klase modela,  
a) i b) prikazuju pogled ptičjeg oka dok c) i d) prikazuju pogled sa strane gdje se vidi razlika

[[Ovaj paragraf možda izostaviti? ili dalje doraditi/premjestiti ili obraditi kasnije kao zasebnu mrežu]]Jedno moguće rješenje ovog problema je uvođenje „druge kamere“, tj. dodati još jedan unos u neuronsku mrežu[[izvor](https://www.kaggle.com/joosthazelzet/one-versus-two-camera-setup-to-recognize-lego/log)]. Ako mreža ima pristup dvjema slikama istog modela, može postići bolju točnost prilikom predviđanja modela u slikama. Mrežu će na ovaj način biti teže koristiti u pravom životu ako je dostupna samo jedna kamera (npr. od mobilnog uređaja), ali lakše ako postoji stroj za specifično ovu svrhu

Ako se uzme u obzir nesigurnost predviđanja mreže iz prošlog paragrafa, očekuje se predviđena maksimalna točnost od oko 85 do 90% za 25 klasa, te najbolje aktualno trenirane mreže[west] su postigle točnost od oko 90%, što podupire raspon postotka.

Gore spomenuti raspon postotaka je razlog poteškoća uspoređivanja mreže sa ostalim neuronskim mrežama, zajedno sa problemom generalne klasifikacije: mreža iz rada klasificira specifične modele LEGO kocaka, dok mreže iz općih radova klasificiraju nad nekim poznatim podatkovnim skupovima kao što su *imagenet* [[izvor](https://image-net.org/)]. Time ostale mreže mogu najčešće klasificirati LEGO kocku samo kao „LEGO kocka“, umjesto „LEGO 3003 2x2 kocka“.

[[Praktični kreirani strojevi i LEGO radovi]]

Ipak se može usporediti u nekim pogledima modificirana VGG16 mreža sa ostalim mrežama iz istog područja, kao što je mreža Jacques Mattheija [[izvor](https://jacquesmattheij.com/sorting-lego-many-questions-and-this-is-what-the-result-looks-like/#accuracy)] gdje se mreža koristi da klasificira tip modela (jeli je dana slika modela kocke, ograde, *Technic* bloka, vegetacije, itd.). Na 60000 primjera podatkovnog skupa za treniranje ima točnost od oko 95%, više za češće modele kao kocke, manje za rjeđe modele kao što su npr. rotacijska ploča (engl. *turntable*). Razlika u točnosti među mrežama dolazi od toga da Mattheijova mreža ne mora točan model odrediti, makar mu se da slika kocke sa strane ili od gore, ono će samo razmišljati kojem skupu pripada i taj tip LEGO kocke sortirati ovisno o rezultatu predviđanja u odgovarajuću košaru (košare za kocke, tanke ploče, široke ploče, vegetaciju, figurice, zastave, itd.).

Još jedan primjer stroja je stroj za sortiranje kojeg je izradio Daniel West [[izvor](https://towardsdatascience.com/a-high-speed-computer-vision-pipeline-for-the-universal-lego-sorting-machine-253f5a690ef4)] iz inspiracije prethodno spomenutog Mattheijovog stroja. West spominje da je njegov stroj „sljedeći korak u evoluciji“, po tome što ima sposobnost prepoznavanja LEGO kocaka po modelu. To znanje se koristi u stroju za sortiranje tako da se modeli grupiraju po tipu objekta i zajedno svi spreme (slično kao Jacquesov stroj) u zajedničke kante (kocke u kocke, grede sa gredama, komadi figurica sa ostalim komadima, itd.). Uspjeh stroja leži u načinu kako je koncipiran. Originalno je treniran nad sintetičkim podatcima, te kad mu se daju stvarne slike iz pravog svijeta, ima lošu točnost predviđanja. Koristeći nasumično unaprjeđenje skupa podataka, West je prestao pokušavati generirati što sličniju sliku realnome nego je krenuo generirati slike koje se više koncentriraju na specifične detalje modela. Sa povećanjem točnosti na kraju je West počeo koristiti bilježenje oznake uz pomoć njegove mreže, mreža mu je ponudila što ono misli da je najbliže predviđanje pa West mora odabrati koji je to točno model, te na tom odabiru mreža se uči. West tvrdi da je oko 100,000 slika uspješno označio na ovaj način. Jedini problem je što West nije objavio nikakve podatke, metrike ili označene slike u javnosti, te zbog toga se ne može pravilno uspoređivati sa našom mrežom, iako postoje razni artikli i par videozapisa koje je West postavio iz čega se vidi efektivnost stroja[[izvor](https://www.youtube.com/watch?v=04JkdHEX3Yk)].

Rad koji je napisan u Delft sveučilištu tehnologije[[izvor](https://arxiv.org/pdf/2008.01584.pdf)] se bavi sa usporedbom LIME i Grad-CAM metoda pojašnjavanja te vuče inspiraciju od stroja Daniel Westa. U radu se spominju metode pojašnjavanja i razumijevanja dubokih neuronskih mreža nad primjerom mreže za klasifikaciju slika LEGO kocaka[west]. [[realno nisam siguran dali je ovaj rad dobar za state of the art, nastavit cu pisati ako mentor kaze ok]]

[[Teorijski radovi]]

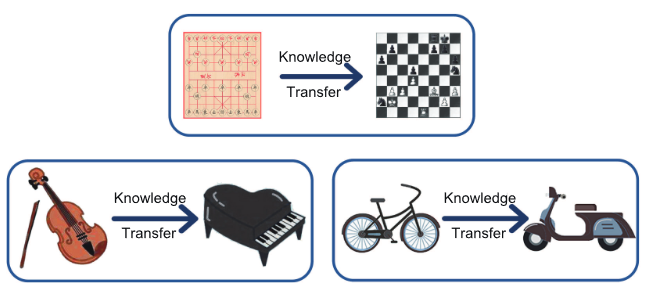
[[Teorijski radovi – data augmentation]]

Rad napisan u Gdansk Sveučilištu tehnologije [[izvor](https://www.researchgate.net/profile/Agnieszka-Mikolajczyk-3/publication/325920702_Data_augmentation_for_improving_deep_learning_in_image_classification_problem/links/5d5d5569458515210257607c/Data-augmentation-for-improving-deep-learning-in-image-classification-problem.pdf)] opisuje razne metode i primjene proširenja skupa podataka za poboljšanje dobivenih rezultata mreža za duboko učenje. Rad se temelji na dvije vrste proširenja skupa podataka, iskrivljavanje podataka i [[oversampling]]. Iskrivljavanje podataka se osniva na korištenju transformacija kao što su afine transformacije: translacija, rotacija, refleksija i smicanje sa nasumičnom razinom učinka (npr. translacija se može izvesti samo u rasponu ±20%, smicanje samo ±15°, itd.), te korištenje transformacija nad kanalima boje u slici kao što su izjednačavanje histograma, mijenjanje kontrasta i mijenjanje „topline“ bijele boje. Još neke dodatne transformacije koje se mogu izvesti su izoštravanje i pomućenje slike. [[oversampling]] se osniva na generiranju sasvim novih podataka učeći uzorke od postojećih ulaznih podataka. Jedna od tih tehnika [[oversampling]] se naziva GAN[[prijevod?]] (engl. *Generative Adversarial Network*), GAN, i općenito [[oversampling]] nije optimalan za mreže koje trebaju prepoznavati značajke slične LEGO kockama jer one ne mogu [[ili im je vrlo teško bez neke vanjske pomoći]] pravilno naučiti generirati umjetne podatke modela koji su strogo povezani sa svojim osima (npr. pravokutnike, trokute, kocke) ili strogom pozicijom teksture na modelu (u našem slučaju pozicija dugmadi LEGO kocke na vrhu kocke, ne može 2x2 kocka imati 6 dugmadi). Iz ovog rada se koristi generalno znanje da čak sa malim učinkom proširenja skupa podataka se poboljšavaju rezultati generalizacije dubokih mreža te dodaje motivaciju za istraživanje optimalnog skupa parametara s kojim se mogu postići bolji rezultati.

Dodatan rad koji opisuje proširenje skupa podataka su napisali korisnici *Journal of Big Data* (Dnevnik velikih podataka) Connor Shorten i Taghi M. Khoshgoftaar [[izvor](https://link.springer.com/content/pdf/10.1186/s40537-019-0197-0.pdf)], te slično prethodno navedenom radu opisuju probleme nedostataka ulaznih podataka za treniranje dubokih mreža (npr. u polju medicine) te istražuju moguća rješenja. Isto tako se dijeli na iskrivljavanje podataka i [[oversampling]], u kontekstu iskrivljavanja podataka veliku pažnju rad obraća na sigurnost primjene neke transformacije nad ulaznim podatcima (sigurnost u smislu da ulazni podatak ne izgubi svoju konekciju nad oznakom koju nosi, npr. u MNIST[[izvor](http://yann.lecun.com/exdb/mnist/)] skupu podataka za rukom pisane brojeve, važno je ne primjenjivati refleksiju, te držati rotaciju unutar prihvatljivih granica, inače se može dobiti pobuna oko nekih brojeva kao 6 i 9 ako se previše rotira[izvor str. 8]). Još jedan problem koji opisuje je kad su ulazni podatci savršeno centrirani, duboke mreže učene nad takvim podatcima zahtijevaju da su novi podatci za predviđanje isto savršeno centrirani, sličan problem se nalazi u ovom radu jer sve generirane slike su centrirane, pa translacijom se taj problem do neke razine uklanja (npr. imat će i dalje poteškoće prepoznavati modele na samom rubu slike). Još jedna tehnika koju ovaj rad primjenjuje od spomenutog rada jest zamućivanje pozadina ulaznih slika, jedan od problema koji mogu nastati korištenjem ove duboke mreže jest slikanje, naprimjer mobitelom, neke LEGO kocke. Sama LEGO kocka će ostati oštra ali pozadina neće biti u fokusu te biti će zamućena. U ovom radu se pozadinske slike lagano zamućuju prilikom učenja da bi se simulirao gubitak fokusa pozadine.

[[Teorijski radovi – transfer learning radovi]]

Rad iz Instituta računalne tehnologije Beijing [[izvor](https://arxiv.org/pdf/1911.02685.pdf)] opisuje prijenosno učenje u znanstvenom polju dubokih mreža.[[dali treba sada opisati šta je prijenosno učenje ili opisati u 4. poglavlju ili negdje ranije?]] Prijenosno učenje je koncept prenošenja korisnih informacija iz jedne domene učene duboke mreže u drugu domenu koja sadrži visok stupanj korelacije, naprimjer iz mreže koja je učena prepoznavati oblike prenesemo težine na mrežu koja uči LEGO kocke, zato što postoje naučeni identifikatori rubova učenje nove mreže će se ubrzati. Ovaj postupak se može izvoditi samo među mrežama sa sličnim ulaznim domenama (po slici 3.2., kao što su: vožnja bicikla se može prenijeti na vožnju motora, sviranje instrumenta svoje obitelji, kao što je flauta, na sviranje instrumenta sličnog tipa, kao klarinet, itd.).



**Slika 3.2.** – Prikaz mogućnosti prijenosa znanja za potrebe prijenosnog učenja [[izvor](https://arxiv.org/pdf/1911.02685.pdf#figure.1)]

Rad opisuje prednosti prijenosnog učenja, gdje ako nova mreža ima manjak ulaznih podataka na kojima se može učiti da koristeći prijenosne mreže može pomoći ispuniti tu početnu prazninu treniranja, iako manjak ulaznih podataka i dalje smeta razvoju nove mreže jer mora razbiti učenje u zadnjim konvolucijskim blokovima prenošene mreže da bi bila efektivna. Jedan od problema prijenosnog učenja je događaj „negativnog prijenosa“ [[izvor](https://openaccess.thecvf.com/content_CVPR_2019/papers/Wang_Characterizing_and_Avoiding_Negative_Transfer_CVPR_2019_paper.pdf)], gdje prethodno naučeno znanje mreže smeta učenju nove mreže umjesto da pomogne. Ovo je rijedak događaj ali uvijek je moguće kada postoji manjak označenih podataka za učenje.

Za ovaj rad to ne bi trebao biti problem jer označenih podataka postoji blizu beskonačno, jer uvijek se mogu generirati novi podatci. Ovaj rad koristi prijenosno učenje na takav način da duboke neuronske mreže VGG16 i VGG19 [[izvor](https://arxiv.org/pdf/1409.1556.pdf(2014.pdf)] sa učenim težinama nad skupom podataka *ImageNet* se učitaju. Te se duboke neuronske mreže promijene na takav način da se odbaci njihov konačan blok slojeva potpuno povezanih neurona koje služe kao klasifikator mreže, te na tom mjestu se kreira novi blok slojeva potpuno povezanih neurona koji će služiti kao novi klasifikator. Na ovaj način se efektivno prenose naučene konvolucijske blokove stare mreže (koje znaju npr. u nižim razinama prepoznavati rubove, u višim razinama prepoznavati oblike) te učenjem nad novim podatcima blok klasifikatora se uči nad novim podatcima uz pomoć već naučenih blokova starih mreža. Na početku treninga je potrebno smrznuti konvolucijske blokove, inače nasumične težine u novom bloku klasifikatora naprave velike promjene u stanju težina konvolucijskih blokova, time uništavajući unaprijed trenirane značajke (otkrivanje rubova, oblika, itd.).

### Poteškoće kod sintetičkih podataka

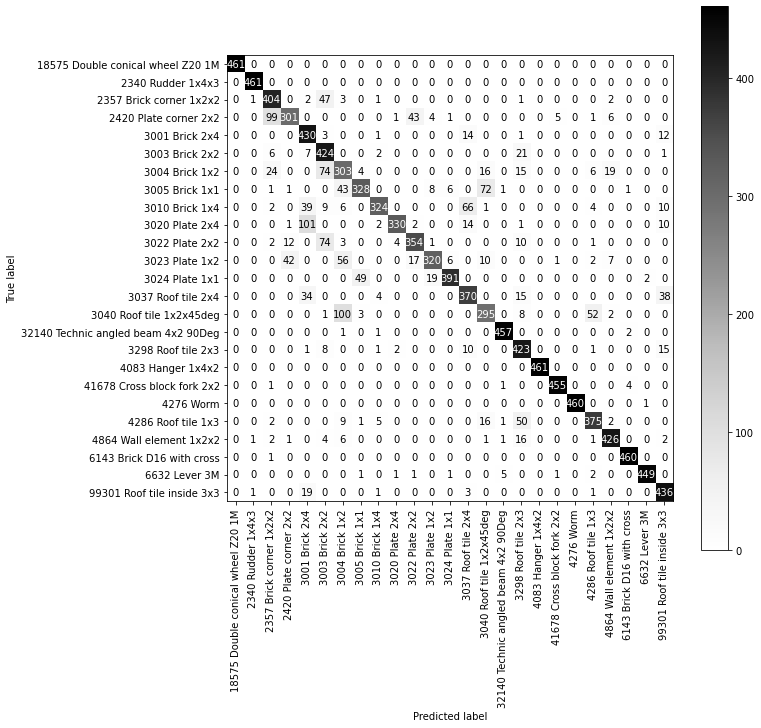
Umjetne neuronske mreže koje su trenirane nad čisto sintetičkim slikama često nailaze na problem generalizacije kad se pokuša dobiti predviđanje nad pravim slikama[[izvor](http://proceedings.mlr.press/v119/chen20x/chen20x.pdf)]. Takvi problemi najčešće dolaze od nedostatka „realnosti“ u sintetičkim slikama tijekom generiranja, primjeri nedostataka su: nerealistične teksture površina(problem odsjaja, problem materijala, problem boje materijala), problem položaja svjetala, problem scene tijekom generiranja, te razni drugi problemi. Rezultati tih problema su trenirane mreže koje su se pretrenirale nad sintetičkim podatcima, te ne znaju reagirati na moguće promjene koje dolaze od pravih slika (npr. u vezi LEGO kocaka, kocke mogu biti oštećene ogrebotinama, mogu biti prozirne, može se mreži dati kocka na kojoj je trenirana ali kocka ima obojane detalje ili teksturu kao što su slike, brojeve, itd.)

Slike generirane u *Blender*-u su prozirne slike, što znači da koristeći biblioteke za manipuliranje slikama (npr. *opencv*) možemo te slike staviti na nasumične pozadine. Time mreža ne mora trenirati na čisto crnoj pozadini, što uklanja problem kada LEGO kocka u predviđanju ima neočekivanu pozadinu što zbuni mrežu prilikom predviđanja. Pozadina se mora sastojati od neupadljivih boja sa što manje oštrih kutova, razlog tome je što mreža može pokupiti detalje pozadine i njih koristiti za odlučivanje modela LEGO kocke. Najkvalitetnije pozadine za taj slučaj bi bile sve vrste pozadina koje su nejednake (npr. izgledaju kao šum) po cjelini, bez uočljivih oblika ili ako imaju oblike da su minimalne, tj. da su točke ili neuočljivi rubovi. Neki primjeri pravog svijeta bi bile slike krečenih zidova (jednolika boja, mekane točke, mekane sjene), jednostavan papir i karton, površine teksture stiropora (sječeni stiropor sadrži puno malih krugova, ako se mreža trenira nad time zajedno sa drugim pozadinama, sadržavat će bolju generalizaciju).

[[confusion matrix, stanje feature mapa? rasporediti negdje drugdje sadržaj?]]

[[matrica zabune]]

Iz matrice zabune na najboljem modelu za slučaj ovog rada (slika 3.3.) se može jasno vidjeti greška zabune gdje umjetna neuronska mreža pogrešno odabere klasu jer je preslična iz nekog kuta nekoj drugoj klasi.



**Slika 3.3.** – Matrica zabune za umjetnu neuronsku mrežu baziranu preko prijenosnog učenja na VGG16 mreži

Iz matrice zabune se vide sva podudaranja među pravim klasama (vertikalna os) i predviđenim klasama (horizontalna os), što se može iskoristiti za istraživanje odnosa podudaranja među klasama. Neki od očitih odnosa se vide iz klasa koje imaju neku sličnu značajku, kao što su 4286 1x3 kocka za krov i 3298 2x3 kocka za krov, jedina razlika među modelima je širina kocaka.

Neke očekivane sličnosti od prethodno spomenutog objašnjenja zašto mreža ne može postići bolje rezultate (blizu početka podpoglavlja 3.1.) ne vrijede ili nisu očekivane po ovoj matrici zabune, naprimjer 2x4 kocka ima jako mali broj podudaranja sa drugim sličnim klasama LEGO kocaka, dok 2x4 ploča ima visok broj podudaranja sa drugim sličnim klasama kao što su 2x4 kocka, očekivalo bi se da je broj pogrešnih podudaranja sličan za te dvije klase. Na ovaj način se iz matrice očitava da je umjetna neuronska mreža osjetljiva tek na sličnosti među klasama (tj. modelima LEGO kocaka), te da nasumična pozadina ne utječe toliko jako na njenu sposobnost predviđanja klase.

[[jeli je dobar paragraf?]]Jedan mogući problem matrice zabune je u tome što ono prikazuje predviđanje sa najvećim postotkom vjerojatnosti, ne radi sa svim decimalnim težinama za neki bolji prikaz sličnosti među klasama, iako ono tek toliko podataka može strpati u prikladan prostor

# [[PYTHON KOD, potrebno?, tu sam kontao transfer learning obraditi]]

[[U ovom dijelu rada će se objasniti neka korištenja *Python* programskog kôda.]]

# ZAKLJUČAK

# PRIZNANJA

Ovaj rad nije sponzoriran sa strane tvrtke LEGO™, niti je primio dopuštenje od LEGO™ za korištenje njihovih modela kocaka.

Po pravilima poštene igre (engl. *fair play*) LEGO™ zahtjeva da se na neautoriziranim radovima ne koristi LEGO™ logo, te da se riječ „LEGO“ uvijek koristi kao pridjev (npr. „napravljeno od LEGO kocaka“, ne „napravljeno od LEGO-a“).

# LITERATURA

# SAŽETAK

# ABSTRACT

# PRILOG A: Popis korištenih modela i njihovi prikazi

|  |  |  |  |
| --- | --- | --- | --- |
| 2340 Rudder 1x4x3 |  | 3040 Roof tile 1x2 45° |  |
| 2357 Brick corner 1x2x2 |  | 3298 Roof tile 2x3 33° |  |
| 2420 Plate corner 2x2 |  | 4083 Hanger 1x4x2 |  |
| 3001 Brick 2x4 |  | 4276 Worm |  |
| 3003 Brick 2x2 |  | 4286 roof tile 1x3 33° |  |
| 3004 Brick 1x2 |  | 4864 Wall element 1x2x2 |  |
| 3005 Brick 1x1 |  | 6143 Brick D16 with cross |  |
| 3010 Brick 1x4 |  | 6632 Lever 3M |  |
| 3020 Plate 2x4 |  | 18575 Double conical wheel Z20 1M |  |
| 3022 Plate 2x2 |  | 32140 Technic angled beam 4x2 90° |  |
| 3023 Plate 1x2 |  | 41678 Cross block fork 2x2 |  |
| 3024 Plate 1x1 |  | 99301 Roof tile inside 3x3 33° |  |
| 3037 Roof tile 2x4 |  |  |  |