

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U
OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

Sveučilišni studij

**PRIMJENA NEURONSKIH MREŽA U
KLASIFICIRANJU LEGO KOCAKA**

Diplomski rad

Anto Tufeković

Osijek, 2021.

SADRŽAJ

1. UVOD	1
1.1. Ukratko o umjetnim neuronskim mrežama, klasifikaciji i problemu prikupljanja podataka	2
1.2. Odabran skup podataka	5
1.3. Pregled rada.....	5
2. PREGLED PODRUČJA RADA.....	6
3. SUSTAV ZA IZRADU KLASIFIKATORA LEGO KOCAKA.....	10
3.1. Implementacija sustava za generiranje podataka	10
3.1.1 Generiranje podataka	10
3.2. Usporedba postojećih skupova podataka	12
3.2.1 Generirani skup podataka i rezultati	12
3.3. Implementacija ANN za klasifikaciju	15
3.3.1. Opis slojeva i strukture	17
3.4. Usporedba s drugim umjetnim neuronskim mrežama.....	25
4. DEMONSTRACIJA I ISPITIVANJE FUNKCIONALNOSTI SUSTAVA	27
4.1. Demonstriranje mogućnosti sustava.....	27
4.2. Ispitivanje funkcionalnosti sustava	29
4.3. Analiza rezultata.....	31
5. ZAKLJUČAK	34
PRIZNANJA.....	36
LITERATURA	37
SAŽETAK	40
ABSTRACT.....	41
PRILOG A: Popis korištenih modela i njihovi prikazi	42

1. UVOD

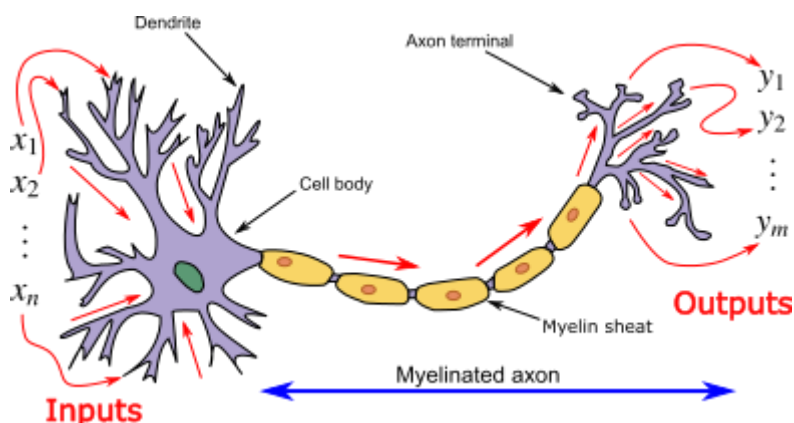
Često se moraju odrediti točni detalji o nekom objektu ili stvorenju, primjerice psa ili mačke gdje vlasnik ne zna dovoljno o pasminama da utvrdi točno kojoj pripada. Za taj problem mogu poslužiti umjetne neuronske mreže koje su trenirane nad više tisuća različitih slika svih vrsta pasmina domaćih životinja 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 jabuka i krušaka), vrste glava šarafa te vrste i modeli igrački, koji su objekt promatranja u ovom radu.

Jedna od situacija gdje ova tehnologija može pomoći je prilikom dizajniranja npr. mobilne aplikacije za slijepe ili slabovidne osobe, osobe s poteškoćama vida i osjetila dodira (svi formalni lijekovi sadrže ime u Brailleovom pismu) mogu iskoristiti ovakve tehnologije za npr. prepoznavanje kutije lijekova po njenim karakteristikama (veličina kutije, boja kutije, šare po kutiji, prepoznavanje teksta na kutiji) da odaberu točne lijekove u pripadajućem trenutku.

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 s 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.

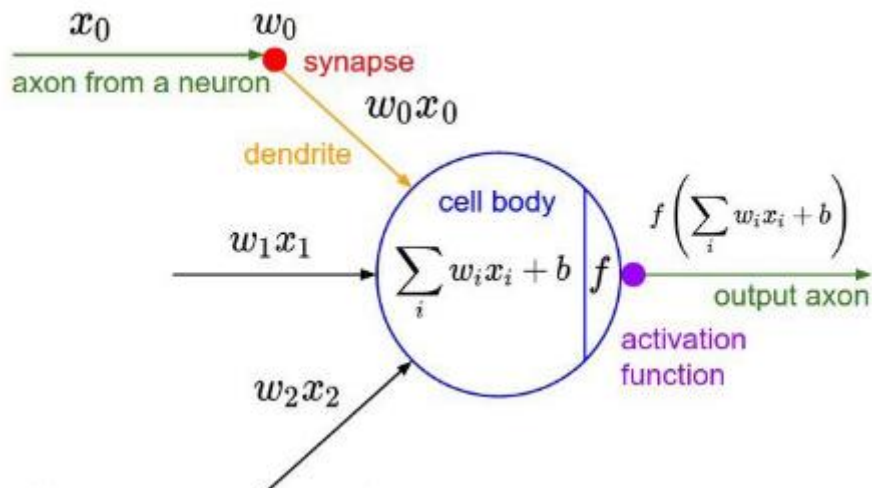
1.1. Ukratko o umjetnim neuronskim mrežama, klasifikaciji i problemu prikupljanja podataka

U ovom radu će se koristiti **duboke umjetne neuronske mreže** (engl. *deep artificial neural networks*) za potrebe korištenja u algoritmu klasifikatora. Umjetne neuronske mreže su vrsta algoritma koji su bazirani nad neuronskim mrežama životinja, te njihova osnovna građevinska jedinica (neuron) ispada vrlo sličan biološkom neuronu kao na slici 1.1.



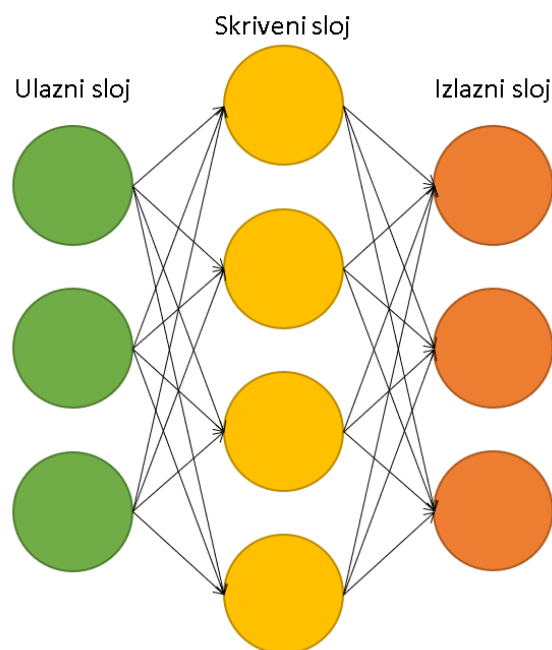
Slika 1.1. Uzor neurona je prava biološka živčana stanica [1]

Na slici 1.2. se vidi da svaki neuron po uzoru akson – sinapsa – dendrit (izlaz – prijenos – ulaz) ima svoje ulaze (x_1, x_2, \dots) na dendritima, neku funkciju stanice u tijelu, te onda preko aksona ima izlaz(y_1, y_2, \dots) kojim se signal dalje propagira po sustavu. Prethodno spomenuta funkcija stanice neurona određuje kakav će signal 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 1.2. Matematički model neurona [2]

Osnovni matematički model neurona je da svaki ulaz ima svoju težinu [3], 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.



Slika 1.3. Primjer neuronske mreže s jednim skrivenim slojem

Kao što je prethodno spomenuto, neuroni su često poslagani u slojevima zbog jednostavnosti kreiranja modela, no 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.

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 (naprimjer osoba može imati krv tipa „A“, „AB“, „A+“, itd.), mogu biti bazirane na brojevima (naprimjer slike imaju piksele koje sadrže tri kanala za boje, najčešće s vrijednostima od 0 do 255) ili jednostavno uspoređivati dolazeće podatke s prijašnje treniranim podacima 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“.

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 podacima 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 podacima 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[4]. Za kreiranje nasumične kopije originala često se koriste jednostavne funkcije kao što su affine transformacije (rotacija, skaliranje, translacija, refleksija, smicanje), transformacije nad histogramom (npr. ujednačenje), te mijenjanje svojstva slike kao što su kontrast, svjetlina ili transformacije kao povećanje oštine, pomućenje slike i izjednačavanje svjetline.

Razlog za proširenjem skupa podataka je poboljšanje generalizacijskog svojstva mreže i povećanjem dostupnog broja uzoraka podataka. Ako se kompleksna mreža trenira nad malim brojem podataka za normalan broj epoha, dogodit će se podtreniranje mreže jer mreža nije

dovoljno trenirana da uspješno prepozna objekt na slici, ali ako se trenira nad malim brojem podataka za više epoha mreža će početi pamtiti ulazne podatke i dogodit će se pretreniranje što opet smeta generalizaciji, jer pri treniranju će imati visoku točnost, a pri validiranju nisku točnost.

1.2. Odabran skup podataka

Ovaj rad je usredotočen na prepoznavanje slika LEGO kocaka. Svaki model LEGO kocke se postavlja kao klasa (npr. 3001 model i 3003 model su sami svoje klase) te kad bi se ciljali svi modeli svih setova ikad izdanih, postojalo bi bezbroj klasa. Za potrebe ovog rada te za potrebe iskazivanja mogućih problema perspektive uzima se u obzir 25 odabranih modela kao klase ovog rada. Svi modeli i njihovi brojevi modela i prikazi se nalaze u prilogu A.

Odabrane su LEGO kocke jer one sadržavaju jednostavne značajke nad kojima bi se duboke neuronske mreže trebale moći trenirati. Treniranje se svodi na sintetički generiranim podacima, te tu se uvodi problem generalizacije i razlika u domenama realnih i sintetičkih podataka.

1.3. Pregled rada

Rad sadrži 3 glavna poglavlja i zaključak. Prvo glavno poglavlje se bavi opisom trenutnog stanja aktualnih radova (engl. *state of the art*) gdje se opisuju neka postojeća rješenja za LEGO klasifikaciju, te dodatni radovi znanstvene zajednice koji su vezani za ovaj rad. Drugo glavno poglavlje se bavi generiranjem podataka za ovaj rad te pojedinostima vezano za to i bavi se s implementiranjem dubokih umjetnih neuronskih mreža koristeći pripadajuće alate (Python, Keras, itd.). Četvrto poglavlje sadrži demonstraciju i ispitivanje svojstava mreže. U zaključku se nalaze izlazne riječi autora s mišljenjima o radu, rezultatima rada, doprinosu te prednostima i manama mreže ovog rada.

2. PREGLED PODRUČJA RADA

Klasificiranje slika su ljudi izvodili već duže vrijeme, AlexNet [5] 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 (engl. *top-5 error rate*), najviše jer su koristili treniranje preko grafičkih kartica. Ovo je privuklo globalnu pozornost prema koristi područja strojnog 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 s 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 [6]).

Joseph Redmon iz Washington sveučilišta je koautor rada [7] gdje opisuje svoju mrežu YOLO9000, koja u stvarnom vremenu može detektirati i klasificirati objekte s preko 9000 kategorija klasa. U radu spominje razna poboljšanja za originalni YOLO model za detekciju te time ide dalje sa YOLOv2 kojeg iskušava na skupovima podataka PASCAL VOC [8] i COCO [9]. Na PASCAL VOC model radi na 67 okvira po sekundi i dobiva performanse od oko 76.8 mAP, dok na COCO na 40 okvira po sekundi dobiva performanse od oko 78.6 mAP.

Modificirana VGG16 mreža Jacquesa Mattheija [10] je mreža koja se koristi da klasificira tip modela LEGO kocke na slici (jeli je dana slika modela kocke, ograde, *Technic* bloka, vegetacije, daje tip, ne direktno model.). 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.) koristeći mehanički stroj s pokretnom trakom. Stroj polako ubacuje pojedine kocke, koje onda računalo uslika, predvidi mu klasu i koristeći mlaznice potisnutog zraka usmjerava kocke u točnu košaru.

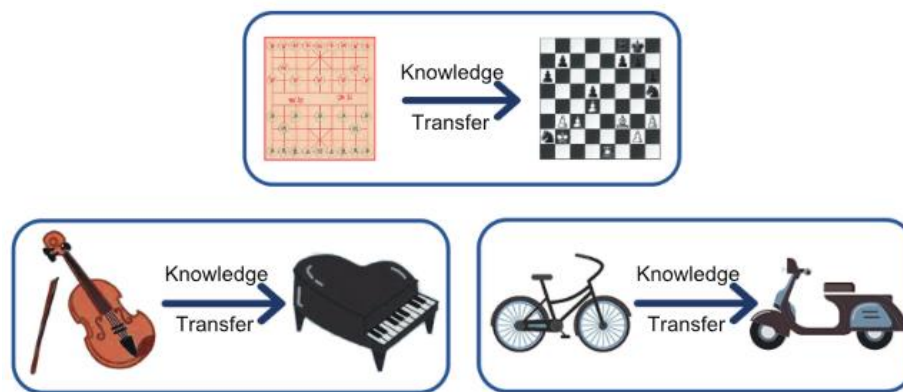
Još jedan primjer stroja je stroj za sortiranje kojeg je izradio Daniel West [11] iz inspiracije prethodno spomenutog Mattheijovog stroja. West spominje da je njegov stroj „sljedeći korak u evoluciji“, po tome što ima sposobnost prepoznavanja LEGO kocaka po modelu. To znanje

se koristi u stroju za sortiranje tako da se modeli grupiraju po tipu objekta i zajedno svi spremaju (slično kao Jacquesov stroj) u zajedničke kante (kocke u kocke, grede s gredama, komadi figurica s ostalim komadima, itd.). Uspjeh stroja leži u načinu kako je koncipiran. Originalno je treniran nad sintetičkim podacima, 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 u pokušaju smanjivanja razlike domena stvarnih i sintetičkih slika. S 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 (nadzirano učenje). 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 s našom mrežom, iako postoje razni artikli i par videozapisa [12] koje je West postavio iz čega se vidi efektivnost stroja.

Rad kojeg je napisala Agnieszka Mikołajczyk u Gdansk Sveučilištu tehnologije [4] 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 korištenje drugih mreža za generiranje podataka (engl. *oversampling*). Iskrivljavanje podataka se osniva na korištenju transformacija kao što su afine transformacije: translacija, rotacija, refleksija i smicanje s nasumičnom razinom učinka (npr. translacija se može izvesti samo u rasponu $\pm 20\%$, smicanje samo $\pm 15^\circ$, 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 *oversamplinga* se naziva GAN (engl. *Generative Adversarial Network*). GAN i općenito *oversampling* nisu optimalne za mreže koje trebaju prepoznavati značajke slične LEGO kockama jer one ne mogu pravilno naučiti generirati umjetne podatke modela koji su strogo povezani sa svojim osima (npr. pravokutnike, trokute, kocke) ili strogom pozicijom tekture 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 s 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 [13], 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 podacima (sigurnost u smislu da ulazni podatak ne izgubi svoju konekciju nad oznakom koju nosi, npr. u MNIST [14] 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[13, str. 8]). Još jedan problem koji opisuje je kad su ulazni podatci savršeno centrirani, duboke mreže učene nad takvim podacima 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 bit će zamućena. U ovom radu se pozadinske slike lagano zamućuju prilikom učenja da bi se simulirao gubitak fokusa pozadine.

Rad Fuzhen Zhuang iz Instituta računalne tehnologije Beijing [15] opisuje prijenosno učenje u znanstvenom polju dubokih mreža. 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 2.1., 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 2.1. – Prikaz mogućnosti prijenosa znanja za potrebe prijenosnog učenja [15 str.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“ [16], 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.

U ovom radu se koriste razni koncepti opisani u prethodno navedenim radovima, zbog manjka raznolikosti podataka (postoji malen broj skupova podataka LEGO kocaka, generirani skup je baziran nad skupom od korisnika Joost Hazelzet sa stranice Kaggle [17]) se koristi proširenje skupa podatka koristeći tradicionalne metode (opisane u gore spomenuta dva rada kao afine transformacije), te koristi se prijenosno učenje za svrhe istraživanja tih mogućnosti, što će se kasnije ispostaviti kao najbolje rješenje u ovom radu.

3. SUSTAV ZA IZRADU KLASIFIKATORA LEGO KOCAKA

3.1. Implementacija sustava za generiranje podataka

Blender [18] je program 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:

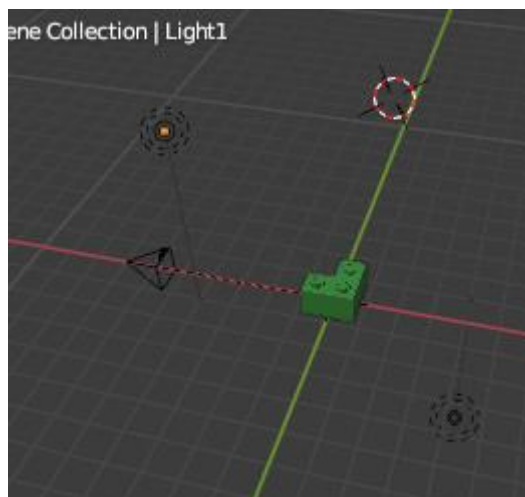
https://github.com/ATufekovic/LEGO_classification

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 s 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. Ovo stoji tu za znatiželjne koji žele pregledati sve vezano za rješenje ovog rada.

Na *github* poveznici se ne nalazi dobiveni skup podataka niti dobivena mreža zbog ograničenja veličine (*github* se ne bi trebao koristiti kao repozitorij skupova podataka i jako velikih datoteka).

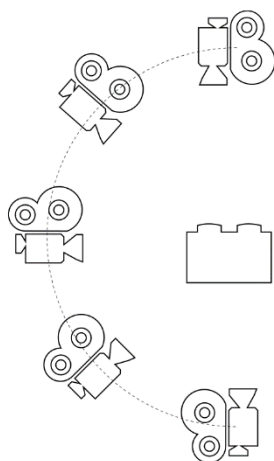
3.1.1 Generiranje podataka

U *Blenderu* je potrebno učitati gore spomenutu datoteku scene, unutar nje se nalaze svi potrebni modeli ovog rada, sve što korisnik mora je prenijeti modele iz nevidljive skupine u vidljivu i obratno kako je potrebno. Sintetički podatci za ovaj rad su generirani u *Blenderu* koristeći skriptu (nalazi se zajedno sa scenom) koja je namještena da uslika zadani model iz raznih kutova. Te slike imaju dva svjetlosna izvora u blizu sredine ali su za jednu prostornu jedinicu odmaknute svaka u svoju stranu (na slici 3.1. se vidi da su odmaknute od centra 4 prostorne jedinice, u stranu pomaknute za 1 prostornu jedinicu te visoko/duboko postavljeno 6 prostornih jedinica).



Slika 3.1. Isječak iz Blender prozora za prikaz lokacija objekta, svjetla i kamere

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 s 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 3.2. Prikaz načina kretanja kamere u Blenderu

Kamera slika model počevši od skroz gore (slika 3.2.), te u zadanim koracima se kreće sferično oko modela prema dolje. Kako prolazi tako kamera „slika“ model, te generirane slike spremne za daljnje korištenje u treniranju. Kamera se tijekom kretanja ne rotira.

Za rad su korišteni modeli koji su ilustrirani u Prilogu A. Cilj je bio koristiti modele koji imaju raznolike stupnjeve sličnosti, više o tome u potpoglavlju 3.2.1.

3.2. Usporedba postojećih skupova podataka

Na internetu je uvijek izazov pronaći kvalitetan skup podataka za svrhu nečijeg projekta ili rada. Jedan od izazova spomenutih u podpoglavlju 1.1. u vezi nedostatka podataka za treniranje se veže dalje ovdje. Naime, ako postoji nedostatak potrebno je ili dobiti više podataka iz nekog izvora (npr. više slika iz područja medicine, kao što su slike magnetske rezonancije MRI, podatci dobiveni iz EKG-a pacijenata sa specifičnim bolestima, RTG slike kostiju za svrhe otkrivanja bolesti, itd.) ili je potrebno te podatke „krivotvoriti“ ali u obliku da su što bliže originalnim podacima u svrsi, tj. potrebno je iskoristiti proširenje skupa podataka ili sintetičko generiranje.

Umjetni podatci imaju široki raspon svrha za koje se mogu koristiti [19] te najčešće pojednostavljaju 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 s 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 s pomućenjem u nekom smjeru, kao da je osoba koja je slikala micala kameru pri slikanju).

Jedan od sintetičkih skupova podataka na internetu je od prethodno spomenutog Joost Hazelzet sa stranice Kaggle [17]. Njegov skup sadrži 50 klasa što je jednako 50 modela LEGO kocaka, gdje je svaki model uslikan 800 puta iz raznih kutova. Jedan od problema sa Hazelzetovim skupom jest da su sve slike neprozirne. Neprozirnim slikama je teže dodati prozirnost u svrhe generalizacije, te zbog toga je odlučeno kreirati novi skup podataka. Nad Hazelzetovim skupom podataka su neki Kaggle korisnici napravili mreže za klasifikaciju, kao što je DenseNet201 mreža korisnika stpete_ishii na [20] s validacijskom točnošću od 66,35% i MobileNet mreža korisnika Datalira na [21] s validacijskom točnošću od 85,69%.

Još jedan skup podataka jest od prethodno spomenutog Daniel Westa, West tvrdi da je kreirao skup podataka od preko 100 000 slika prema [22], iako taj skup nije javno objavljen.

3.2.1 Generirani skup podataka i rezultati

U prethodna dva poglavlja su pojašnjene koristi sintetički generiranih podataka, za slučaj u radu se zna da LEGO kocke imaju oštro definirane oblike (nikada se dvije kocke istog

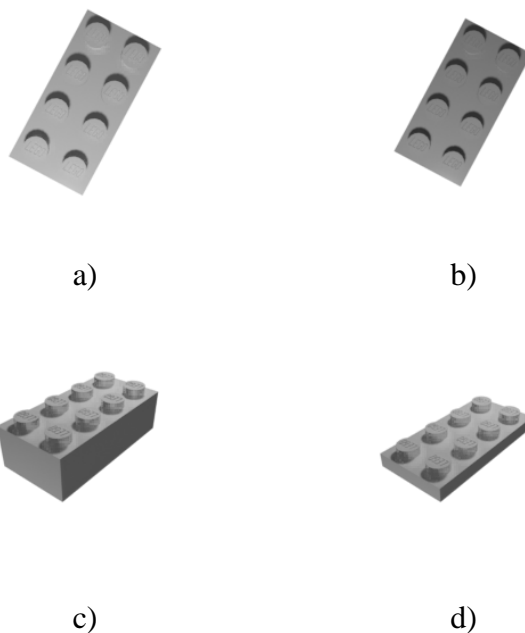
modela ne razlikuju, osim možda po boji), kocke imaju rubove i dugmadi, ploče imaju glatke površine bez dugmadi, štapići za povezivanje su cilindričnog oblika sa specifičnim reljefima na krajevima, kotači su široki cilindri s reljefima na kontaktnoj površini, itd.

Na gore spomenutoj github stranici rada se nalaze *Jupyter* bilježnice s modelima, te njihove strukture i rezultati preko ispisa u bilježnicama.

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 podacima jednostavne mreže mogu postići visok rezultat kao što je vidljivo na tablici 3.1. Problem na koji se nalazi 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 3.3.). 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 s potpunom sigurnošću reći koji je to model osim ako nije slikano dovoljno u stranu, da se vidi druga strana modela. Sličan koncept se može primijeniti i 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 mreža napravi 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 (dali je kocka ili ploča).



Slika 3.3. 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. Jedno moguće rješenje ovog problema je uvođenje „druge kamere“, tj. dodati još jedan unos u neuronsku mrežu [23]. 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 (naprimjer 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 [22] su postigle točnost od oko 90% (relativna točnost na više od 25 klasa), što podupire raspon postotka.

3.3. Implementacija ANN za klasifikaciju

U radu se koriste mreže VGG16 i VGG19 [24] za svrhe prijenosnog učenja. Aktivacijske funkcije umjetnih neurona spomenutih u potpoglavlju 1.1. su sve *ReLU* funkcije koje su često korištene u praksi. *ReLU* (engl. *Rectified Linear Unit*, ispravljena linearna jedinica) i njena derivacija sadrže sljedeći oblik:

$$f(x) = \max(0, x), \quad f'(x) = \begin{cases} 0 & \text{ako } x < 0 \\ 1 & \text{ako } x > 0 \\ \text{neodređeno} & \text{ako } x = 0 \end{cases} \quad (3-1)$$

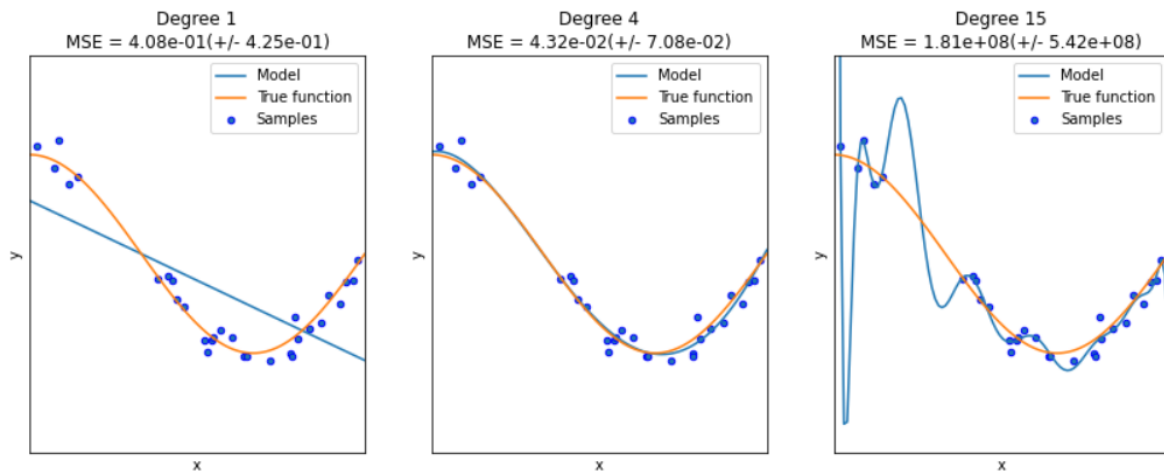
gdje x je ulazna vrijednost, a $f(x)$ izlazna vrijednost funkcije.

Aktivacijska funkcija mora biti derivabilna zbog svrhe treniranja algoritmom povratnog razmnožavanja (engl. *backpropagation algorithm*) [25]. Na kraju je potrebna optimizacijska funkcija, te odabran je Adam algoritam, koji je oblik algoritma stohastičkog gradijentnog spusta iliti SGD [26]. Adam algoritam je nadogradnja *RMSProp* [27] algoritma na takav način da ukomponira svojstva *AdaGrad* [28] 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 s 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 spremati 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 3.4. Grafički prikaz za utjecaj parametara treninga na pretreniranje [29]

Na slici 3.4. se vide primjeri tri modela s drugačijim parametrima treniranja. Prvi model je oblika polinomna regresija s jednim stupnjem, zbog toga se ne može točno podacima namjestiti (engl. *underfit*). Drugi model ima četiri stupnja, što je dostatno da se model namjesti trening podacima 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 sposobnosti 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 s 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.

3.3.1. Opis slojeva i strukture

Prvo je potrebno učitati podatke zajedno sa svim parametrima vezano za proširenje skupa podataka kao što je prikazano u slici 3.5.

Linija Kod

```
1:     import numpy as np #prvo potrebni uvozi
2:     import keras
3:     import matplotlib.pyplot as plt
4:     import cv2
5:     from keras.applications.vgg16 import VGG16
6:     from keras.preprocessing.image import ImageDataGenerator
7:     from keras.models import Model
8:     dataset_path = "../..//LEGO_brick_custom/"
9:     dataset_output = "./output"
10:    bg = Background_source("../..//background_images", (200,200),
(3,3))#kreiranje objekta za postavljanje pozadinskih slika
11:    train_datagen = ImageDataGenerator(
        shear_range=10,
        zoom_range=[1,1.2],
        height_shift_range=0.1,
        width_shift_range=0.1,
        brightness_range=[0.6,1],
        rotation_range=90,
        horizontal_flip=True,
        vertical_flip=True,
        preprocessing_function = bg.apply_background_preprocess_input
    )
12:    validation_datagen = ImageDataGenerator(
        preprocessing_function = bg.apply_background_preprocess_input
    )
13:    train_generator =
    train_datagen.flow_from_directory(os.path.join(dataset_output, 'train')
    color_mode="rgba", target_size=(200,200), shuffle=True,
    class_mode="categorical")
14:    validation_generator =
    validation_datagen.flow_from_directory(os.path.join(dataset_output,
    'val'), color_mode="rgba", target_size=(200,200), shuffle=True,
    class_mode='categorical')
```

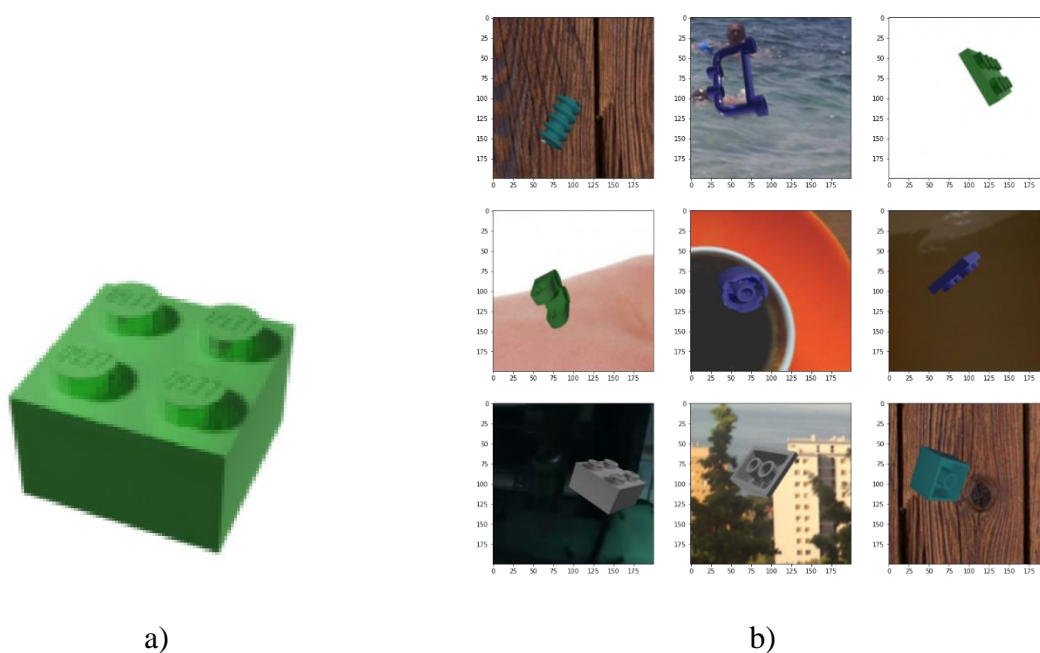
Slika 3.5. Blok koda za uvoz podataka i označavanje vrste proširenja skupa podataka

U kodu su odabrani sljedeći parametri u vezi proširenja skupa podataka:

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

- Nasumična refleksija horizontalno i vertikalno
- Nasumična rotacija, raspon od 0° do 90°

Koristeći refleksiju u oba smjera zajedno s nasumičnom rotacijom od 90° , pokrivene su sve mogućnosti pozicije nekog LEGO modela te dobivaju se rezultati s prihvatljivim izgledom nad osnovnim slikama s prozirnošću kao na slici 3.6. a). Koristeći ovakve postavke dobivaju se slike kao po slici 3.6. b) s nasumičnom pozadinom za bolju generalizaciju i smanjivanje razlike u domenama.



Slika 3.6. a) Slika modela 3003 LEGO kocke, b) Slika modela LEGO kocaka proširene raznim tehnikama

Odabrana najbolja mreža po tablici 3.1. je duboka umjetna neuronska mreža bazirana na prijenosnom učenju VGG16 mreže. Mreža je kreirana po slici 3.7.

Linija Kod

```
1:     from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout, Lambda, Input
2:     def remove_alpha(image):
3:         return image[:,:,:,:-1]
4:
5:     new_input = Input(shape=(200,200,4))
6:     lam = Lambda(remove_alpha, output_shape=(200,200,3))(new_input)
7:     temp = Model(inputs=new_input, outputs=lam)
8:
9:     base_model = VGG16(weights='imagenet', include_top=False,
input_tensor=temp.output)
10:    for layer in base_model.layers:
11:        layer.trainable = False
```

Slika 3.7. Blok koda za uvoz i kreiranje VGG16 arhitekture

Zato što je arhitektura VGG16 osnovana na neprozirnim slikama, a u radu se rukuje sa slikama koje sadrže četvrti kanal, alfa kanal za prozirnost, potrebno je na samom ulazu maknuti taj alfa kanal (slika 3.7. linija 2, linija 6) koji se dobiva iz generatora (slika 3.5.) da bi sve pravilno radilo s postojećim težinama VGG16 mreže jer alfa kanal je svejedno napunjen samo jedinicama, nema vrijedne značajke u sebi.

Prijenosno učenje se izvodi tako da se parametar *include_top* u osmoj liniji slike 3.7. postavi na *false* dok se težine preuzmu s interneta s *weights='imagenet'*. Nakon toga je potrebno postaviti novi blok gusto spojenih neurona (slika 3.8.) da služi kao klasifikator mreže, ono bi koristilo izlaze konvolucijskih blokova da bi odlučilo o kakvoj kocki na slici se radi.

Linija Kod

```
1:     top_model = base_model.output
2:     top_model = Flatten()(top_model)
3:     top_model = Dense(1024)(top_model)
4:     top_model = Dropout(0.5)(top_model)
5:     top_model = Dense(512)(top_model)
6:     top_model = Dense(len(train_generator.class_indices),
activation="softmax")(top_model)
7:
8:     model = Model(inputs = base_model.inputs, outputs = top_model)
```

Slika 3.8. Blok koda za kreiranje novog bloka neurona za klasifikaciju

U slici 3.8. se vidi kod za postavljanje odgovarajućih slojeva koji čine novi blok potpuno spojenih umjetnih neurona. Sloj ispada (engl. *dropout*) se koristi za regularizaciju, jednostavno rečeno ono služi kao most između dva sloja, te ovisno o zadanom postotku toliko posto konекcija blokira tijekom jedne epohe treniranja. Po [30] smanjuje se pretreniranje te ne utječe toliko jako na ostatak mreže.

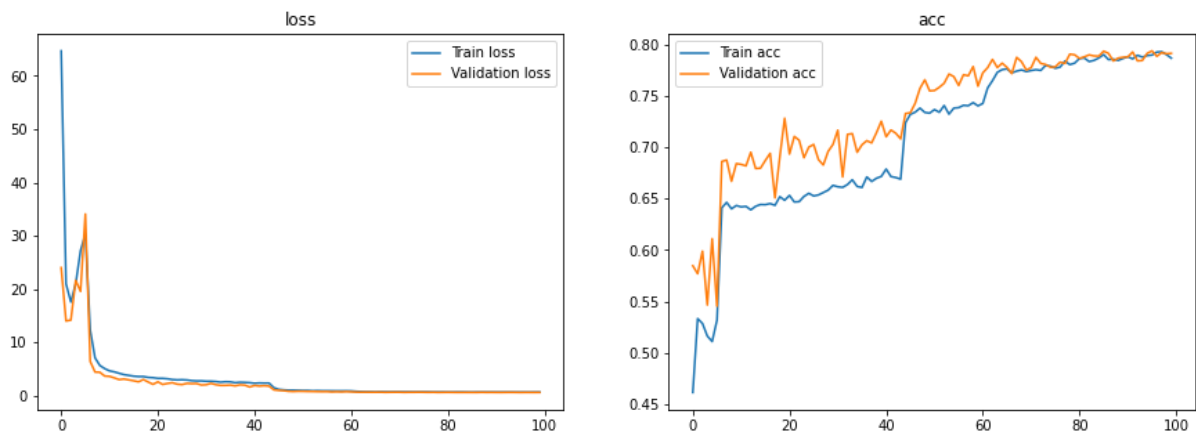
Na kraju je potrebno izvesti treniranje u dva koraka, prvi puta sa smrznutim slojevima originalne mreže (provedeno u slici 3.7.), jer inače da su ti slojevi otključani, nasumične težine u novom gusto spojenom bloku bi uzrokovale razne i po mogućnosti štetna ažuriranja težina u konvolucijskim blokovima, što može slijediti do uništenja postojećih detektora rubova i oblika.

Linija Kod

```
1:      from keras.optimizers import Adam
2:      opt = Adam(lr=1e-3)
3:      model.compile(
4:          loss='categorical_crossentropy',
5:          optimizer=opt,
6:          metrics=['accuracy']
7:      )
8:
9:      history = model.fit_generator(
10:         train_generator,
11:         validation_data = validation_generator,
12:         epochs = 100,
13:         verbose = 1,
14:         callbacks=callbacks
15:     )
```

Slika 3.9. Blok koda za prvo treniranje

Tijekom treniranja u Jupyter bilježnici se postavlja ispis. Time se može pratiti trenutačno stanje treniranja, te na kraju epohe prikazuje vrijednost funkcije gubitka i točnosti tijekom treniranja i validacije. Po tim metrikama se mjeri kvaliteta mreže. Nakon prvog treniranja se može pregledati *history* varijabla preko grafičkog ispisa.



Slika 3.10. Grafički prikaz metrika preko epoha tijekom prvog treniranja

Po grafu na slici 3.10. se vidi da su pravilno odabrani parametri za stopu učenja u Adam optimizacijskom algoritmu, iako u treniranju sudjeluje i adaptivno mijenjanje stope učenja (ako se ne dobiju bolji rezultati metrika unutar 4 epohe, smanji stopu učenja na 25% trenutne vrijednosti) što polako vodi Adam algoritam prema boljim rezultatima. Drugo učenje se provodi tako da se omogući treniranje nad zadnjim konvolucijskim blokom.

Linija Kod

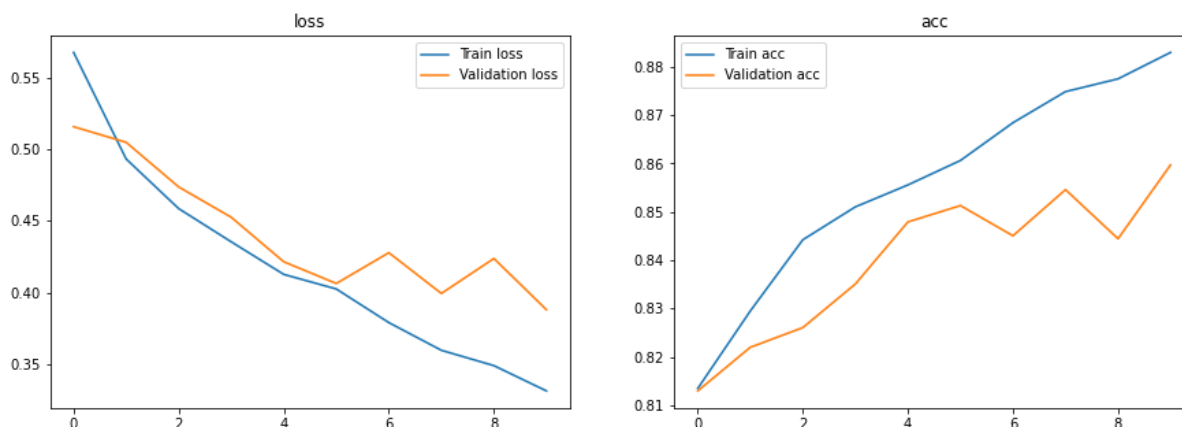
```

1:     for layer in base_model.layers[:-4]:
2:         layer.trainable = True
3:     opt = Adam(lr=1e-6)
4:     model.compile(
5:         loss='categorical_crossentropy',
6:         optimizer=opt,
7:         metrics=['accuracy']
8:     )
9:     history = model.fit_generator(
10:         train_generator,
11:         validation_data = validation_generator,
12:         epochs = 10,
13:         verbose = 1,
14:         callbacks=callbacks
15:     )

```

Slika 3.11. Blok koda za drugo treniranje

Potrebno je odmrznuti zadnja četiri sloja (kao na slici 3.11.), te postaviti optimizacijski algoritam na nisku početnu vrijednost. Dobiveni rezultati su grafički prikazani na slici 3.12.



Slika 3.12. Grafički prikaz metrika preko epoha tijekom drugog treniranja

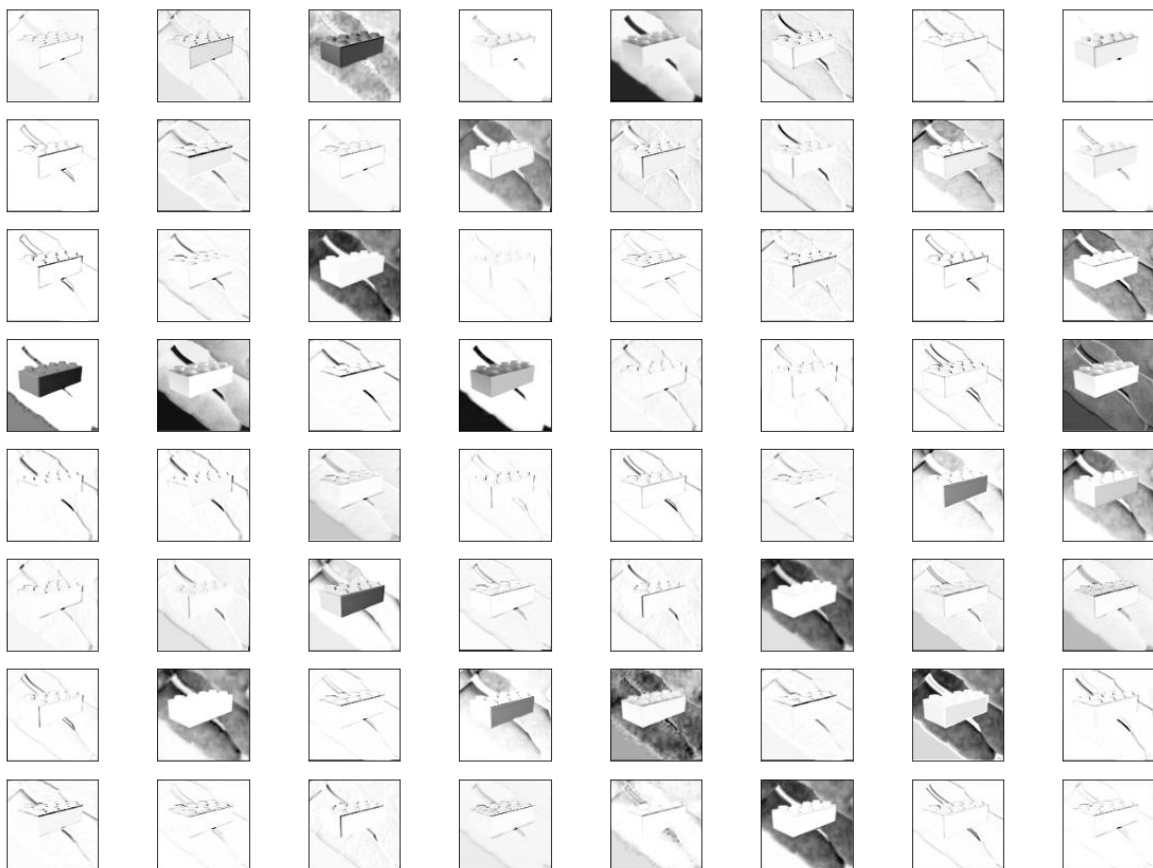
Vidljivo je poboljšanje jer ovaj puta je blok gusto povezanih neurona naviknut na izlaz konvolucijskih mreža, te neće odmah uništiti konvolucijski blok, nego se daje prilika konvolucijskom bloku da se privikne na izlaze prethodnih konvolucijskih blokova u slučaju učenja nad LEGO kockama.

Matrice zabune su grafički prikaz svih predviđanja koje je neka mreža izvela [31]. U ovom slučaju gdje imamo umjetnu neuronsku mrežu s više klasa, dobiti će se matrica zabune s $N \times N$ polja, gdje N stoji za broj klasa (te time redaka i stupaca). U vertikalnoj osi svaki redak predstavlja istinitu oznaku (engl. *true label*), dok na horizontalnoj osi svaki stupac predstavlja klasu s njenim predviđenim brojem pogodaka (engl. *predicted label*). „Cilj“ mreže bi bio što više polja u dijagonali „pogoditi“ jer ta polja su istiniti pozitivni (engl. *true positive*), dok svi ostali su lažni pozitivni (engl. *false positive*) zbog pogrešno predviđene klase.

Iz matrice zabune na najboljem modelu za slučaj ovog rada (slika 3.13.) se može jasno vidjeti greška zabune gdje umjetna neuronska mreža pogrešno odabere klasu jer je previše slična iz nekog kuta nekoj drugoj klasi.

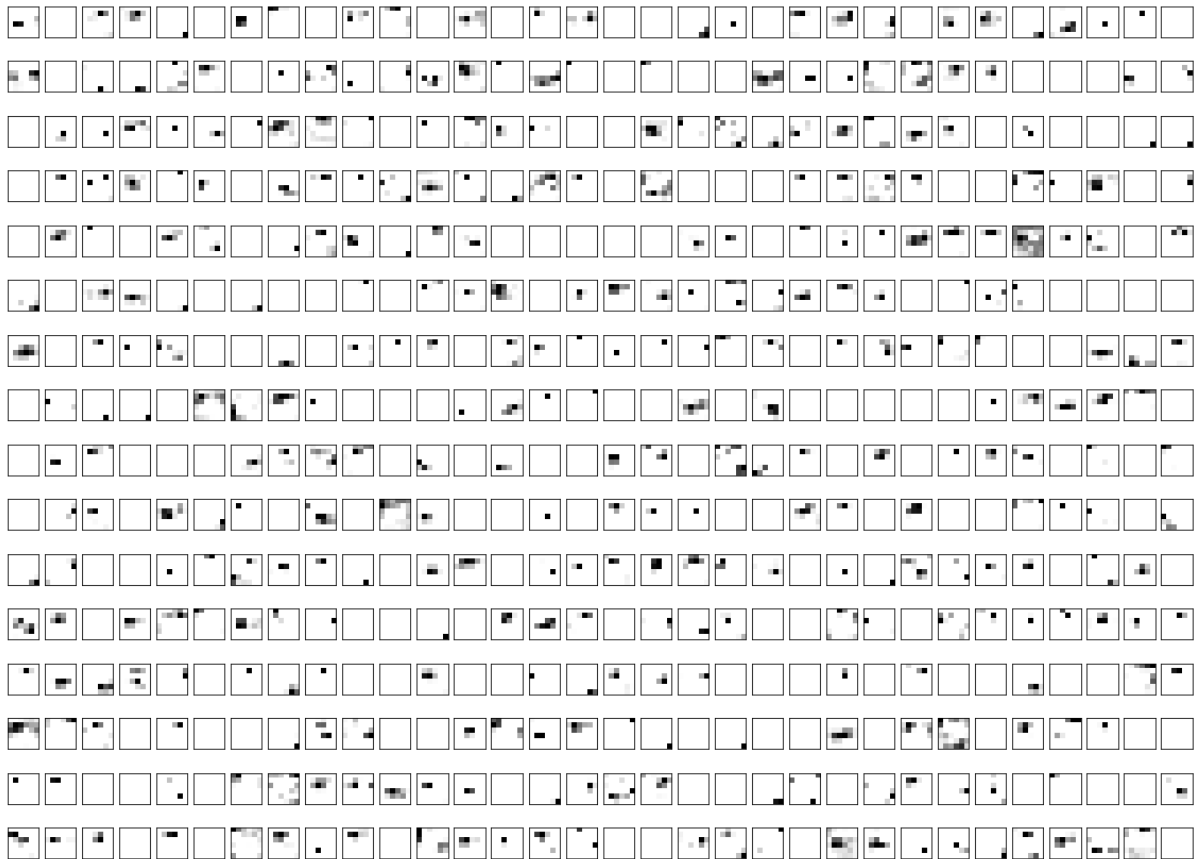
očekivalo bi se da je broj pogrešnih podudaranja sličan za te dvije klase. Na ovaj način se iz matrice očitava da je umjetna neuronska mreža osjetljiva tek na sličnosti među klasama (tj. modelima LEGO kocaka), te da nasumična pozadina ne utječe toliko jako na njenu sposobnost predviđanja klase (tj. ne utječe negativno), što je dobro jer pomaže pri generalizaciji umjetne neuronske mreže. Jedan mogući problem matrice zabune je u tome što ono prikazuje predviđanje s najvećim postotkom vjerojatnosti, ne radi sa svim decimalnim težinama za neki bolji prikaz sličnosti među klasama, iako ono tek toliko podataka može staviti u prikladan prostor da bude prihvatljiva vidljivost podataka.

Još jedan dodatan alat za analizu kvalitete umjetne neuronske mreže je prikaz slika značajki (engl. *feature maps*). Slike značajki se dobivaju na izlazu filtra jednog sloja koji je djelovao na ulaz iz prethodnog sloja, najčešće se uzimaju iz izlaza konvolucijskih slojeva da bi dobili prikaz toga što oni vide i na što im se izlazi aktiviraju kao što je prikazano na slici 3.14.



Slika 3.14. Prikaz slika značajki na izlazu prvog konvolucijskog sloja

U promatranoj umjetnoj neuronskoj mreži se samo zadnji konvolucijski blok trenira, i to na kraju treninga joj se otključa sposobnost treniranja na 10 epoha. Međutim, izlaz tog konvolucijskog bloka je 512 slika značajki s rezolucijom 6x6 piksela, iz čega se ne može vidjeti značajna informacija ali je dobro vidjeti ikakvu aktivaciju nad slikama filtera.



Slika 3.15. Prikaz slika značajki na izlazu zadnjeg konvolucijskog sloja

Koristeći informaciju dobivenu iz svih konvolucijskih blokova koristeći njihove slike značajki, moguće je zaključiti da umjetna neuronska mreža sadrži dobru sposobnost otkrivanja potrebnih rubova i oblika koje je naslijedila od VGG16 mreže, te aktivnost u zadnja dva konvolucijska bloka dokazuje da konvolucija prethodnih blokova izvlači potrebne značajke da se tim blokovima aktiviraju izlazi te time daju potrebne podatke za predviđanje u bloku slojeva potpuno povezanih neurona.

3.4. Usporedba s drugim umjetnim neuronskim mrežama

Stvorena umjetna neuronska mreža za prepoznavanje LEGO kocaka se može koristiti samo sa originalnim Python okruženjem u kojem je napravljen (najviše zbog specifičnih verzija *Keras* i *Tensorflow* biblioteka koje su korištene za *PlaidML* sloj abstrakcije). Zbog toga postoje poteškoće sa prijenosom mreže među sustavima jer osim spremljene arhitekture i težina slojeva

je potreban popis biblioteka koje *Python* koristi sa točnim verzijama, što je važno držati u umu ako je poželjno testirati mrežu danu u *github* poveznici.

Treniranje dobivene mreže je relativno sporo, ali ne najsporije jer velik dio brzine se veže za to što je blok gusto spojenih neurona manje veličine nego inače, jer mora samo 25 klasa predvidjeti (dok neke druge mreže mogu imati iznad 1000 klasa, što zahtjeva više neurona za kvalitetno predviđanje, naprimjer originalni VGG16 je sadržavao dva sloja od 4096 neurona i konačan sloj od 1000 neurona dok dobivena mreža koristi dva sloja od 1024 i 512 neurona te konačan sloj od 25 neurona). Za treniranje je bilo potrebno 22h za 100 epoha prvog treniranja sa smrznutim konvolucijskim blokovima, te za drugo treniranje je bilo potrebno oko 5.5h za 10 epoha drugog treniranja s odmrznutim zadnjim konvolucijskim blokom, sveukupno oko 27.5h treniranja nad 38 400 slika LEGO kocaka u 25 klasa (~1400 obrađenih ulaza svaki sat, ~0.38 ulaza svake sekunde). Osim toga što sadrži visok broj neurona u mreži, procesor može biti zauzet dok generira proširenje podataka nad ulazom (afine transformacije pa dodavanje pozadine), zahvaljujući paralelizmu između treniranja i generiranja podataka ovaj se utjecaj ne osjeti toliko jako u jačim strojevima koji to mogu podnijeti brže.

Ankita Singh i Pawan Singh iz Indijskog sveučilišta Amity su napravili pregled polja prepoznavanje slika koristeći umjetne neuronske mreže [32]. U radu su opisali neuronske mreže, njihove koristi i korake izrade neuronske mreže. Nad CIFAR-10 [33] skupom podataka su kreirali neuronsku mrežu (10 klasa, 6000 slika svaka gdje su odabrane samo 3 klase: zrakoplovi, ptice i automobili). Umjetna neuronska mreža se zasnivala na tri bloka konvolucije i jedan blok gusto povezanih neurona. Blok gusto povezanih neurona je sadržavao jedan sloj od 3 neurona koji je primao 512 ulaza prethodnog konvolucijskog bloka. Treniranje je naspram dobivene mreže se izvelo puno brže, sa brzinom od 4-5 milisekundi po ulazu, što je ~222 ulaza po sekundi (oko 580 puta brže).

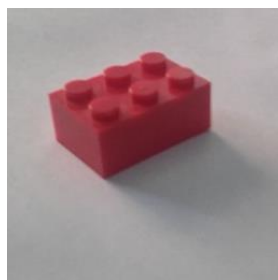
Razlika među mrežama se nalazi u tome što dobivena mreža sadrži masivan broj parametara (utega) koje se moraju ažurirati tijekom treniranja, dok u Singhovoj mreži broj parametara je manji (512 na 3 neurona naspram našeg 18432 na 1024 na 512 na 25 neurona, i to samo u gusto povezanom bloku gdje se nalazi velika većina parametara).

Sličnu priču možemo navesti sa velikim mrežama kao što je originalni VGG16, gdje postoji puno veći broj parametara za svrhe klasifikacije više klasa (18432 na 4096 na 4096 na 1000), gdje će vrijeme treniranja trajati duže nego kod dobivene mreže i puno duže od Singhove mreže.

4. DEMONSTRACIJA I ISPITIVANJE FUNKCIONALNOSTI SUSTAVA

4.1. Demonstriranje mogućnosti sustava

Dobivena duboka umjetna neuronska mreža može nad 200x200 slikama početi prepoznavati objekte ako se ti objekti nalaze unutar prethodno definiranih i treniranih 25 klasa, ne postoji mogućnost pravilnog prepoznavanja ostalih modela LEGO kocaka. Naprimjer ne zna prepoznati model 3002 3x2 kocku jer ju uopće nema kao mogućnost na izlazu, ako joj se da slika tog modela, dobiti će se nasumične vrijednosti predviđanja, često najbližeg modela (npr. dobije se predviđanje slike 4.1. po slici 4.2.).



Slika 4.1. LEGO kocka model 3002

Linija Kod

```
1:     image_path = "../../../test_images/3002.png"
2:     image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
3:     image = bg.apply_background_preprocess_input(image)
4:     image = np.expand_dims(image, axis=0)
5:     prediction = loaded_model.predict(image)
6:     np.set_printoptions(formatter = {'float': lambda x:
7:     "{0:0.3f}".format(x)})
8:     print(prediction[0])
9:     print(prediction[0].tolist().index(max(prediction[0])))
10:    print(list(validation_generator.class_indices.keys())[prediction[0]
11:    .tolist().index(max(prediction[0]))])
...
ispis: [0.00 0.00 0.00 0.00 0.92 0.00 0.00 0.00 0.02 0.00 0.00 0.00 0.00
0.05 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01]
4
3001 Brick 2x4
```

Slika 4.2. Blok koda za pripremu predviđanje i ispis predviđanja

Takva predviđanja dobivena mreža ne može očekivati jer neuroni nisu naučeni na izlaze konvolucijske mreže za LEGO kocku model 3002 („predvidjelo“ je LEGO kocku model 3001). Ako se na sličan način preda slika 2357 modela, za koji je mreža trenirana, dobiju se drugačiji rezultati.



Slika 4.3. LEGO kocka model 2357

Linija Kod

```
ispis: [0.00 0.00 0.36 0.04 0.00 0.32 0.01 0.00 0.01 0.08 0.06 0.00 0.00
0.00 0.00 0.00 0.08 0.00 0.00 0.00 0.00 0.02 0.00 0.00 0.01]
2
2357 Brick corner 1x2x2
```

Slika 4.4. Ispis za predviđanje slike 4.3.

Na slici 4.4. se vidi ispis predviđanja svih klasa te kojoj klasi ono misli da slika pripada, postoji puno mogućih smetnji u slici te zato je predvidjelo s niskim postotkom od 36% da je 1x2x2 rub kocka, a ne sljedeći najbliža 2x2 kocka. Postoji mogućnost da sustavu smeta sjena kocke jer nad trening kockama osim na dugmadima nema sjene, iako opet je ocijenila točno kocku.

Za sliku 4.5. mreža kaže s puno većom pouzdanošću da se radi o LEGO kocki model 3001.



Slika 4.5. LEGO kocka model 3001

Linija Kod

```
ispis: [0.00 0.00 0.00 0.00 0.96 0.00 0.00 0.00 0.01 0.00 0.00 0.00 0.00
0.03 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01]
4
3001 Brick 2x4
```

Slika 4.6. Ispis za predviđanje slike 4.5.

Jedna mogućnost boljeg rada na ovoj slici je razlika objekta i pozadine, sa svijetlim objektom i tamnijom pozadinom, jer dobiva pouzdanost od 96% za klasu 3001. Ovo pravilo ne vrijedi uvijek, jer na sljedećem primjeru je mreža loše ocijenila o kojoj se kocki radi.



Slika 4.7. LEGO kocka model 3003

Linija Kod

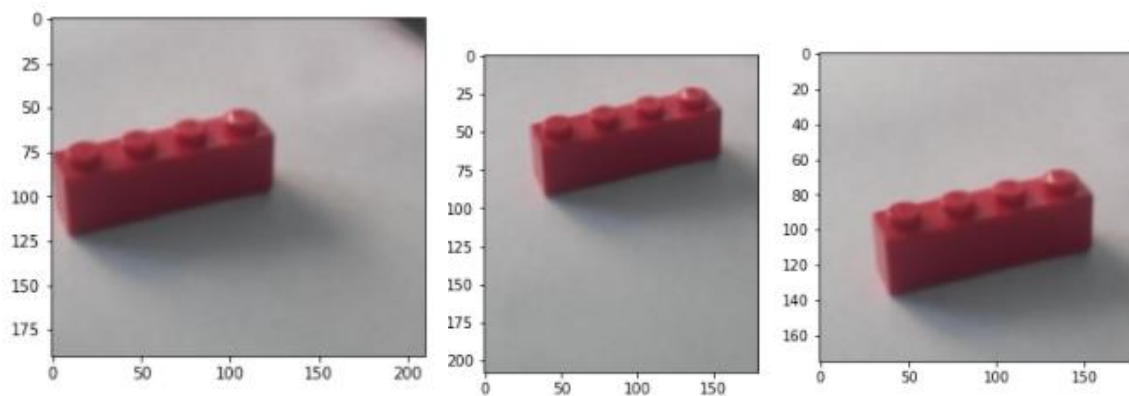
```
ispis: [0.00 0.00 0.00 0.00 0.27 0.12 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.18 0.00 0.00 0.40 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.02]
16
3298 Roof tile 2x3
```

Slika 4.8. Ispis za predviđanje slike 4.7.

Točna klasa je dobila tek 27% vjerojatnosti, dok najviše ocijenjena klasa ima 40%. Razlika od 13% ali kad se gleda najveća vjerojatnost mreža je pogriješila.

4.2. Ispitivanje funkcionalnosti sustava

Dobivena mreža bi, po teoriji, trebala biti otporna na pomake objekta po slici i promijeni boje. Tijekom stvaranja ulaznih slika preko *ImageDataGenerator* u slici 3.5. dani su argumenti za affine transformacije, koje uključuje i pomak horizontalno i vertikalno. Otpornost na boje bi trebalo doći iz činjenice da se koriste nasumične boje u generiranju sintetičkih podataka, u nadi da će dobivena mreža naučiti ignorirati boje i koncentrirati se na rubove i oblike. Zato što su slike generirane takve, ako se mreža uči nad takvim podacima može doći do problema gdje mreža predviđa da objekt gledan pod nekim kutom daje tu boju. U sljedećih par primjera će se istražiti dali je dobivena mreža otporna na takve promjene.



a)

b)

c)

Slika 4.9. LEGO kocka 3010 a) pomaknuta u lijevo, b) pomaknuta prema gore desno, c) pomaknuta prema dolje

Linija Kod

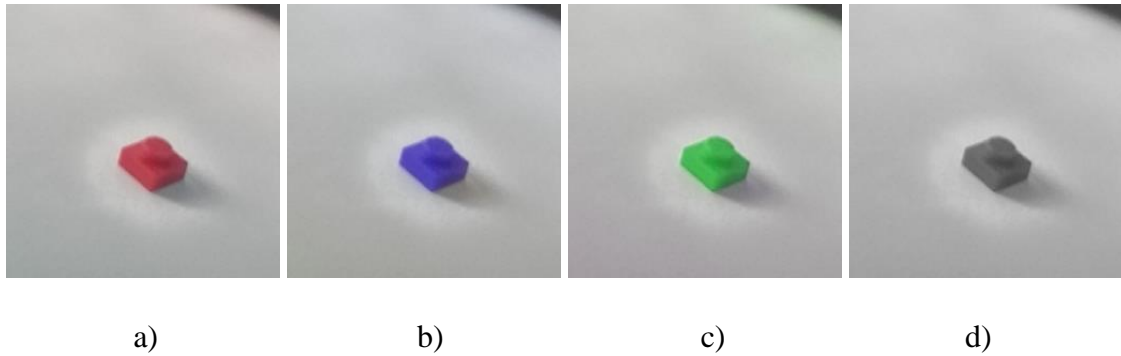
```
ispis      [0.00 0.00 0.00 0.00 0.01 0.00 0.00 0.00 0.98 0.00 0.00 0.00 0.00
a):        0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
8
3010 Brick 1x4
```

```
ispis      [0.00 0.00 0.00 0.00 0.04 0.00 0.00 0.01 0.87 0.00 0.00 0.00 0.03
b):        0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
8
3010 Brick 1x4
```

```
ispis      [0.00 0.00 0.00 0.00 0.08 0.00 0.00 0.00 0.77 0.02 0.00 0.00 0.00
c):        0.13 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00]
8
3010 Brick 1x4
```

Slika 4.10. Ispis za predviđanje slika a), b) i c) u slici 4.9.

Dobivena mreža daje visok postotak vjerojatnosti za pomaknute slike, 98% za sliku 4.9. a), 87% za b) i 77% za c).



Slika 4.11. LEGO kocka 3024 a) originalna boja, b) RGB→BGR, c) RGB→GRB, d) RGB u sive tonove

Linija Kod

```
ispis [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.10 0.00 0.00 0.00 0.02 0.76
a):   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.11 0.00]
12
3024 Plate 1x1
```

```
ispis [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.01 0.42
b):   0.00 0.00 0.00 0.00 0.00 0.01 0.00 0.00 0.00 0.00 0.23 0.00]
12
3024 Plate 1x1
```

```
ispis [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.61 0.00 0.00 0.00 0.01 0.18
c):   0.00 0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.17 0.00]
7
3005 Brick 1x1
```

```
ispis [0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.07 0.00 0.00 0.00 0.05 0.81
d):   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.07 0.00]
12
3024 Plate 1x1
```

Slika 4.12. Ispis za predviđanje slika a), b), c) i d) u slici 4.10.

Za originalnu sliku je dobiven točan rezultat s 76% vjerojatnosti. Mijenjanjem crvenog i plavog kanala dobije se 42% vjerojatnosti ali i dalje točan rezultat. Mijenjanje crvenog i zelenog kanala uzrokuje netočan rezultat gdje točna klasa dobiva samo 18% vjerojatnosti, ali predviđena klasa se razlikuje samo u debljini (predviđena je 1x1 kocka, a ne ploča). Slika u sivim tonovima je dobila najbolji rezultat sa 81% vjerojatnosti predviđanja.

4.3. Analiza rezultata

Sve slike korištene u 4. poglavlju su ručno uslikane. Uslikane su s 1 megapiksela rezolucijom za najgori mogući rezultat (moderne kamere, čak i u mobilnim uređajima podržavaju više od 10 megapiksela inače, što bi dovelo do boljih rezultata). Dobivaju se slike

relativno loše kvalitete s lagano zamućenim rubovima LEGO kocaka, što predstavlja izazov dobivenoj mreži da otkrije odgovarajuće rubove i oblike za predviđanje.

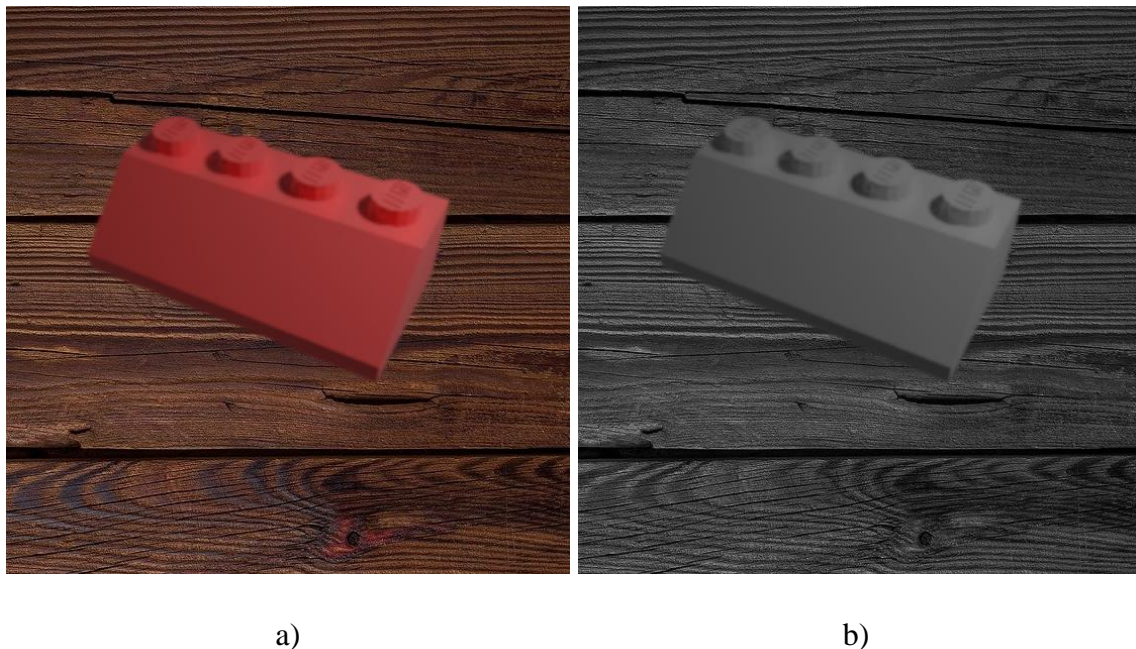
U potpoglavlju 4.1. su predstavljene neke nasumične slike i njihova predviđanja preko dobivene mreže. Vidljivo je odmah nad rezultatima da dobivene vjerojatnosti se ne podudaraju s vjerojatnostima nad sintetičkim podacima, slika 4.5. je postigla zadovoljavajući rezultat, dok slika 4.3. je postigla točan rezultat samo zato što je njena vjerojatnost najveća s 36% dok sljedeća klasa (2x2 kocka) je dobila 32% vjerojatnosti. Kod zapravo LEGO kocke 2x2 dobilo je pogrešan rezultat s time da točna klasa nije ni na drugom mjestu, prvo mjesto je netočna klasa 3298 s 40%, sljedeća klasa je netočna klasa 3001 s 27% vjerojatnosti, sljedeća je opet netočna klasa 3037 s 18%, i tek onda je točna klasa sa samo 12% vjerojatnosti. Moguće da je tu smetala boja LEGO kocke, u podacima za treniranje ne postoje narančaste kocke.

Dalje, u potpoglavlju 4.2. su ispitane neke od jasno vidljivih modela za otpornost mreže nad pomicanjem gledanog objekta i boje identičnog objekta. U prvom ispitivanju je uzeta slika LEGO kocke modela 3010 i pomicana je u tri smjera. U prvoj slici je diralo lijevi rub ali je i dalje dobro predvidjelo klasu LEGO kocke. U drugoj slici je pomaknuto prema gore, s time da se jedan dio slike odsiječe u procesu, to nije upitno za rad mreže jer joj ne smetaju nedostajući podatci na nekim od ulaza (podatci veći od ulaza ruše program). Druga slika je dobila lošije predviđanje, 87% naspram 98% prve slike. Treća slika je pomaknuta prema dolje, isto s odsječenim dijelom, te ona je dala još lošije rezultate sa 77%.

Postoji mogućnost da nedostatak ulaza negativno utječe na predviđanje, ali to nema puno smisla jer prvo treba slika proći kroz 5 blokova konvolucije prije nego li može doći do blokova gusto povezanih neurona. Kod neurona bi smetalo nedostatak ulaza dok konvolucija i dalje radi bez problema jer kako se smanjuje veličina filtra a povećava broj filtara, smanjuje se utjecaj „izgubljenih“ podataka u kasnijim slojevima koji imaju filtre veličine 6x6, gdje je teško primijetiti takav nedostatak podataka.

Konačno ispitivanje je u vezi mijenjanja boja iste slike tijekom predviđanja. Dana je originalna slika LEGO kocke 3024 crvene boje za najveći učinak prilikom mijenjanja kanala boje. Originalna slika dobiva drugi najbolji rezultat od 76% vjerojatnosti, što možda upućuje na to da mijenjanje boje cijele okoline lagano utječe na rezultate predviđanja. Sljedeće dvije slike su dobile 42% i 18%, vezano na mijenjanje boje cijele okoline a ne samo objekta, možda na dobivenu mrežu utječe lagano promijenjena boja papira kojoj su sada isto zamijenjeni kanali boja. Takve sitne razlike nisu uopće vidljive ljudskom oku, ali umjetna neuronska mreža, koja

rukuje s takvim sitnim podacima, može primijetiti te sitne razlike i dobiti utjecaj na predviđanje zbog toga. Zadnja slika gdje je originalna slika pretvorena u sive tonove (svi kanali su iste vrijednosti osim alfa kanala) je dobila i najbolje rezultate od svih danih slika za to ispitivanje s 81% vjerojatnosti. Moguće je da se rezultati mreže mogu poboljšati tako da se sve slike za treniranje pretvore u slike sivih tonova, ali tu postoji mogući problem detekcije rubova po boji, naprimjer: lakše je otkriti crvenu LEGO kocku na smeđem drvenom stolu kao po slici 4.13. a) nego da se slika pretvori u sive tonove što može smetati detekciji točnih rubova (na slici 4.13. b) se vidi da smeđa boja sadrži puno „crvene“, što bi možda uzrokovalo smetnju tijekom detekcije rubova koje bi inače bilo lakše izabrati da postoje sva tri originalna kanala boje).



Slika 4.13. Prikaz koncepta smetnje crvenih objekata nad smeđom pozadinom prilikom pretvaranja slike u sive tonove, a) originalna slika LEGO kocke 3001 na smeđoj pozadini, b) slika pretvorena u sive tonove

Uzevši sve točke dosad navedene u obzir, može se reći da je relativno uspješno prevedena praznina među domenama (engl. *domain gap*) između dobivenih sintetičkih podataka nad kojima se mreža trenirala i realnim uslikanim podacima. Uzevši u obzir da je jedino realno u trening podacima isječene pozadine, a i one su nasumično odabrane što može smetati generalizaciji ako par slika zaredom dobije sličnu pozadinu (iako na 20 slika od preko 1000x1000 rezolucije ta vjerojatnost je malena) dobivena mreža daje zadovoljavajuće rezultate nad stvarnim podacima kad nije nikada bila trenirana nad stvarnim podacima.

5. ZAKLJUČAK

Korištenje dubokih umjetnih neuronskih mreža je učinkovit način za kreiranje klasifikatora nad slikama, gdje važnost objekta kojeg promatramo leži u njenom oštrom obliku i pravilnim teksturama kao kod LEGO kocaka, iako ova spoznaja se može primijeniti i kod drugih sličnih objekata, kao što su kartice pakovanja za lijekove (aluminij na pozadini pakovanja sadrži određenu teksturu i na sebi ima isprintan naziv lijeka i dozu u miligramima, nad time se može duboka mreža učiti).

Za korištenje nad LEGO kockama ispostavlja se da je izvrstan izbor s puno prilagodljivosti s raznim parametrima kojim se može rukovati da se izvede sustav po želji. Primjerice, može se kreirati sustav koji ima više od odabranih 25 klasa ovog rada, kako mogu ostale mreže imati 1000 klasa, može i ova biti trenirana nad više od 25 početno odabranih klasa kao što je Westova mreža.

Za svrhe ovog rada gdje je konačno odabrana duboka umjetna neuronska mreža bazirana nad prijenosnim učenjem od VGG16 mreže kao trenutno rješenje, mreža je dala relativno brzo treniranje za dostupan hardver, te ispunila je očekivane rezultate za kvalitetnim prepoznavanjem danih slika LEGO kocaka.

Postoje moguća poboljšanja sustava:

- kompliciranije mreže se mogu koristiti ako se koristi jači stroj za treniranje, trenutno je RX580 4GB, AMD FX-8350, 8GB DDR3, optimalno bi bilo s više jačih NVIDIA grafičkih kartica većeg VRAM kapaciteta, te procesor istog ranga ili jači s više sustavne memorije, barem 16GB DDR4 ili DDR5
- korištenje mreže koja ima više ulaza, primjerice da sadrži dva ulaza (kao dvije kamere) gdje drugi ulaz prima sliku istog modela ali iz različitog kuta (recimo, 90° odmaknuta po nekoj osi da uslika stranu modela) te time uklanja problem mreže ovog rada i postiže mogućnost dobivanja performansi od više nego 95% točnosti (dok s jednim ulazom s pokojom greškom dobiva maksimalno 85-90%)
- alternativa prethodnog poboljšanja je da se isti objekt snima, naprimjer mobilnom aplikacijom koja detektira model kocke na kameri tako da za svaki model uzme više „uzoraka“ tj. slika te nad njima čini predviđanja. U konačnici se predviđanja mogu pozbrajati te podijeliti s brojem uzoraka da bi se dobila konačna srednja vjerojatnost, u nadi da usmjerava prema jednom modelu

Doprinos ovog rada bi bilo znanje da postoji isplata za ulaganje vremena u istraživanju podataka (u ovom slučaju istraživanje značajki LEGO kocaka, njihovim mogućim problemima i rješenja za te probleme) za prilagodbu dubokih umjetnih neuronskih mreža u koristi klasifikacije sličnih objekata koje mogu biti, naprimjer, druge igračke, elektroničkih komponenata (prepoznavanje vrijednosti otpornika preko slike), dijelova za sastavljanje u automatiziranoj strojarnici i slično. Za ovaj rad postoji već fizičko rješenje u obliku Jacquesseovog i Westovog stroja za sortiranje, te na Kaggle stranici sa skupom podataka Hazelzeta postoji mnoštvo korisnika koji su isprobali i uspješno kreirali neku vrstu mreže za prepoznavanje LEGO kocaka (nepoznate su performanse nad realnim slikama).

Koristeći jednostavne korake (i bolji hardver) svakome je moguće se uputiti u klasifikaciju slika za neki željeni objekt koristeći programska rješenja koje je znanstvena zajednica razvijala zadnjih 20 godina u obliku Pythona, Tensorflowa i Kerasa te tu stoji najveći doprinos ovog rada za korist drugim ljudima.

Konačno, može se reći da ovakvo istraživanje nad specifičnim objektima ima smisla raditi jer je moguće dobiti uvid u unutrašnje funkcioniranje dubokih mreža za neke specifične modele promatranja (npr. LEGO kocke), što može pomoći kod općenitog razumijevanja ostalih vrsta dubokih mreža, te pri razumijevanju procesa treninga dubokih mreža.

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

- [1] L. Vu-Quoc, „File:Neuron3.png“, Wikipedia, 16. rujan 2018., dostupno na: <https://commons.wikimedia.org/wiki/File:Neuron3.png>
- [2] D. Fumo, „A Gentle Introduction To Neural Networks Series — Part 1“, towardsdatascience.com, 4. kolovoz 2017., dostupno na: <https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc>
- [3] K. Hinkelmann, „Neural Networks“, uncam, dostupno na: http://didattica.cs.unicam.it/lib/exe/fetch.php?media=didattica:magistrale:kebi:ay_1718:ke-11_neural_networks.pdf
- [4] A. Mikołajczyk, „Data augmentation for improving deep learning in image classification problem“, Research Gate, svibanj 2018., dostupno na: 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
- [5] A. Krizhevsky, „ImageNet Classification with Deep Convolutional Neural Networks“, papers.nips.cc, 2012., dostupno na: <https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>
- [6] Y. Huang, S. Shakya, T. Odeleye, „Comparing the Functionality between Virtual Reality and Mixed Reality for Architecture and Construction Uses“, davidpublisher.com, 2019., dostupno na: <https://www.davidpublisher.com/Public/uploads/Contribute/5d5e0535ad919.pdf>
- [7] J. Redmon, A. Farhadi, „YOLO9000: Better, Faster, Stronger“, arxiv.org, 2018., dostupno na: <https://arxiv.org/pdf/1612.08242.pdf>
- [8] „GluonCV: a Deep Learning Toolkit for Computer Vision“, cv.gluon.ai, 2021., dostupno na: <https://cv.gluon.ai/contents.html>
- [9] Microsoft, „COCO – Common Objects in Context“, cocodataset.org, 2021., dostupno na: <https://cocodataset.org/#home>
- [10] J. Mattheij, „Sorting 2 Tons of Lego, Many Questions, Results“, jacquesmattheij.com, 28. lipanj 2017., dostupno na: <https://jacquesmattheij.com/sorting-lego-many-questions-and-this-is-what-the-result-looks-like/#accuracy>
- [11] D. West, „A high-speed computer vision pipeline for the universal LEGO sorting machine“, towardsdatascience.com, 1. kolovoz 2019., dostupno na: <https://towardsdatascience.com/a-high-speed-computer-vision-pipeline-for-the-universal-lego-sorting-machine-253f5a690ef4>
- [12] D. West, „The WORLD'S FIRST Universal LEGO Sorting Machine“, youtube.com, 3. prosinac 2019., dostupno na: <https://www.youtube.com/watch?v=04JkdHEX3Yk>
- [13] C. Shorten, T. M. Khoshgoftaar, „A survey on Image Data Augmentation for Deep Learning“, link.springer.com, 2019., dostupno na: <https://link.springer.com/content/pdf/10.1186/s40537-019-0197-0.pdf>

- [14] Y. LeCun, C. Cortes, C. J.C. Burges, „THE MNIST DATABASE of handwritten digits“, yann.lecun.com, 1998., dostupno na: <http://yann.lecun.com/exdb/mnist/>
- [15] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, „A Comprehensive Survey on Transfer Learning“, arxiv.org, 7. srpanj 2020., dostupno na: <https://arxiv.org/pdf/1911.02685.pdf>
- [16] Z. Wang, Z. Dai, B. Póczos, J. Carbonel, „Characterizing and Avoiding Negative Transfer“, arxiv.org, 5. listopad 2019., dostupno na: <https://arxiv.org/pdf/1811.09751.pdf>
- [17] J. Hazelzet, „Images of LEGO Bricks“, „40,000 images of 50 different LEGO bricks“, kaggle.com, 31. prosinac 2019., dostupno na: <https://www.kaggle.com/joosthazelzet/lego-brick-images>
- [18] Blender zaklada, „Open source 3D creation. Free to use for any purpose, forever.“, blender.org, 2021., dostupno na: <https://www.blender.org/>
- [19] C. Dilmegani, „The Ultimate Guide to Synthetic Data in 2021“, research.aimultiple.com, 19. srpanj 2018., dostupno na: <https://research.aimultiple.com/synthetic-data/>
- [20] stpete_ishii, „Lego Brick Classify DenseNet201“, kaggle.com, 11. srpanj 2021., dostupno na: <https://www.kaggle.com/stpeteishii/lego-brick-classify-densenet201/notebook>
- [21] Datalira, „Classify Bricks: Compare Transfer Learning Model“, kaggle.com, 3. lipanj 2021., dostupno na: <https://www.kaggle.com/databeru/classify-bricks-compare-transfer-learning-model#4.-Train-the-architecture-with-the-best-result>
- [22] D. West, „How I created over 100,000 labeled LEGO training images“, towardsdatascience.com, 1. ožujak 2019., dostupno na: <https://towardsdatascience.com/how-i-created-over-100-000-labeled-lego-training-images-ec74191bb4ef>
- [23] J. Hazelzet, „One versus two camera setup to recognize LEGO“, kaggle.com, 3. siječanj 2020., dostupno na: <https://www.kaggle.com/joosthazelzet/one-versus-two-camera-setup-to-recognize-lego/log#Conclusion>
- [24] K. Simonyan, A. Zisserman, „VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION“, arxiv.org, 10. travanj 2015., dostupno na: [https://arxiv.org/pdf/1409.1556.pdf\(2014.pdf](https://arxiv.org/pdf/1409.1556.pdf(2014.pdf)
- [25] Y. LeCun, L. Bottou, G. B. Orr, K-R. Müller, „Efficient BackProp“, yann.lecun.com, 1998., dostupno na: <http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf>
- [26] D. P. Kingma, J. L. Ba, „ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION“, arxiv.org, 30. siječanj 2017., dostupno na: <https://arxiv.org/pdf/1412.6980.pdf>
- [27] G. Hinton, N. Srivastava, K. Swersky, „Lecture 6a Overview of mini-batch gradient descent“, cs.toronto.edu, dostupno na: <http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf>
- [28] J. Duchi, E. Hazan, Y. Singer, „Adaptive Subgradient Methods for Online Learning and Stochastic Optimization“, jmlr.org, 2011., dostupno na: <https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf>

- [29] M. Tripathi, „Underfitting and Overfitting in Machine Learning“, datascience.foundation, 13. srpanj 2020., dostupno na: <https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning>
- [30] N. Sristava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, „Dropout: A Simple Way to Prevent Neural Networks from Overfitting“, jmlr.org, lipanj 2014., dostupno na: <https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf>
- [31] S. V. Stehman, „Selecting and interpreting measures of thematic classification accuracy“, sciencedirect.com, 28. lipanj 1999., dostupno na: [https://doi.org/10.1016/S0034-4257\(97\)00083-7](https://doi.org/10.1016/S0034-4257(97)00083-7)
- [32] A. Singh, P. Singh, „Image Classification: A Survey“, a2zjournals.com, 19. prosinac 2020., dostupno na: https://a2zjournals.com/jieee/uploadpdf/Ankita_Singh_1606489012P49.pdf
- [33] A. Krizhevsky, V. Nair, G. Hinton, „The CIFAR-10 dataset“, cs.toronto.edu, 2009., dostupno na: <https://www.cs.toronto.edu/~kriz/cifar.html>

SAŽETAK

Korištenje dubokih umjetnih neuronskih mreža je postao de-facto standard kao prvi korak pri pokušavanju rješavanja nekog problema vezanog za prepoznavanje slika. Ovakvim načinom se dobije znatna količina znanja koju možemo upotrijebiti za analizu korištenih podataka te za daljnje korake potrebne za poboljšavanje korištene mreže (naprimjer promijene u proširenju skupa podataka ako je preveliko, tipa, smicanje).

U ovom radu su opisane osnove o strojnom učenju, te istraženo je kreiranje raznih mreža nad skupom podataka generiranih u Blender-u. Na kraju je odabrana duboka umjetna neuronska mreža temeljena na prijenosnom učenju od VGG16 mreže. Ta mreža je postigla najbolje rezultate (sitno više nego VGG19 temeljena mreža), te po njoj su istraženi ostali parametri kao što su matrica zabune i njene slike značajke, po prvoj metodi je određeno na kojim modelima mreža ima problema, te po drugoj metodi se otkriva kako mreža vidi dane slike u konvolucijskih blokovima.

Ključne riječi: umjetne neuronske mreže, strojno učenje, LEGO™, proširenje skupa podataka, prijenosno učenje







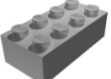
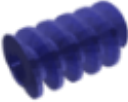
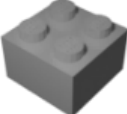
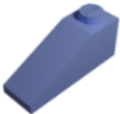
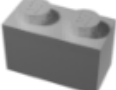





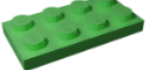



ABSTRACT

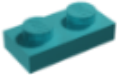



The usage of artificial neural networks has become the de-facto standard as the first step in trying to solve a problem related to image recognition. This way one can obtain knowledge about the problem and use it for further analysis concerning the solution (ie. introducing changes in the data augmentation steps to see how the deep network reacts).

This work explains the basics of machine learning, and it explores the creation of various artificial neural networks which were then trained over an image data set generated in Blender. In the end, the deep artificial neural network based on transfer learning from VGG16 was chosen for its superior characteristics (although the VGG19 based one was close). The network was further explored for its properties, such as its confusion matrix and feature maps, the first research method reveals problematic classes which the network might fail to handle, and the second research method reveals how the network sees the given pictures as the pictures travel down the block of convolution layers.

Key words: artificial neural networks, machine learning, LEGO™, data augmentation, transfer learning

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	