UNIVERZITET U BANJOJ LUCI
ELEKTROTEHNIČKI FAKULTET

Igor Ševo

SPECIJALIZOVANA NEURONSKA MREŽA ZA KLASIFIKACIJU I SEGMENTACIJU AERO-SNIMAKA

DOKTORSKA DISERTACIJA

Banja Luka, 2020.

UNIVERSITY OF BANJA LUKA
FACULTY OF ELECTRICAL ENGINEERING

Igor Ševo

SPECIALIZED NEURAL NETWORK FOR AERIAL IMAGE CLASSIFICATION AND SEGMENTATION

DOCTORAL DISSERTATION

Banja Luka, 2020

Mentor	dr Zdenka Babić, redovni profesor, Elektrotehnički fakultet Univerziteta u Banjoj Luci
Naslov doktorske disertacije	Specijalizovana neuronska mreža za klasifikaciju i segmentaciju aero-snimaka
Rezime	U civilnim i vojnim domenima, za potrebe urbanog planiranja i nadgledanja zemljišta, javlja se potreba za automatskom klasifikacijom i segmentacijom aero-snimaka dobijenih udaljenom detekcijom. Ova disertacija predstavlja rješenje ovog problema korištenjem specijalizovane vještačke neuronske mreže obučene u nadgledanom režimu. Predloženo rješenje je implementirano i postignuti su rezultati uporedivi sa najboljim dosadašnjim u oblasti. Uz rješenje data je i metodologija projektovanja i razvoja neophodna da se ono reprodukuje, te da se izvede implementacija sličnog rješenja u potencijalno različitim oblastima.
Ključne riječi	Mašinsko učenje, neuronska mreža, klasifikacija, segmentacija, aero-snimak, metode projektovanja, metode razvoja
Naučna oblast	Inžinjerstvo i tehnologija
Naučno polje	Elektrotehnika, elektronika i informaciono inžinjerstvo
Klasifikaciona oznaka	T 120
Tip odabrane licence kreativne zajednice	Autorstvo – nekomercijalno – dijeliti pod istim uslovima (CC BY-NC-SA)

Mentor	Dr. Zdenka Babić, full professor, Faculty of Electrical Engineering, University of Banja Luka
Doctoral dissertation title	Specialized neural network for aerial image classification and segmentation
Abstract	In civilian and military domains, for the needs of urban planning and land monitoring, a need arises for automatic classification and segmentation of aerial images obtained by remote detection. This dissertation presents a solution for this problem, utilizing a specialized neural network trained in a supervised mode. The proposed solution is implemented yielding state of the art results. With the solution, the paper presents the methodology required to reproduce the solution, or implement a similar one in different fields of study.
Keywords	Machine learning, neural network, classification, segmentation, aerial image, architectural design methods, development methods
Area of study	Engineering and technology
Field of study	Electrical engineering, electronics and information engineering
Classification label	T 120
Creative commons license type	

Mom djedu.

Mojoj porodici.

SADRŽAJ

1 UVOD

U civilnim i vojnim domenima, sa izuzetnom ekspanzijom tehnologija za akviziciju aero-snimaka, od dronova do bespilotnih letilica sa kamerama visoke rezolucije, pojavljuje se značajna potreba za automatizacijom posla označavanja različitih tipova zemljišta sa ciljem njihove ispravne dodjele za odgovarajuće slučajeve upotrebe. Potencijalne upotrebe zemljišta uključuju urbano planiranje, nadgledanje, monitoring i procjenu usjeva, prevenciju poplava i požara, te je za sve ove oblasti upotrebe od značajne koristi automatizacija procesa prepoznavanja vrste zemljišta i obilježavanja objekata od interesa na snimcima zemljišta. Ova potreba za automatizacijom je uzrokovala pojavu velikog broja metoda za automatsku obradu snimaka zemljišta, prvenstveno aero-snimaka dobijenih udaljenom detekcijom sa letilica, u posljednje vrijeme dominantno onih koji se oslanjaju na neuronske mreže, jer se takvi pristupi pokazuju kao robusniji i generalno bolji u ovim zadacima [1].

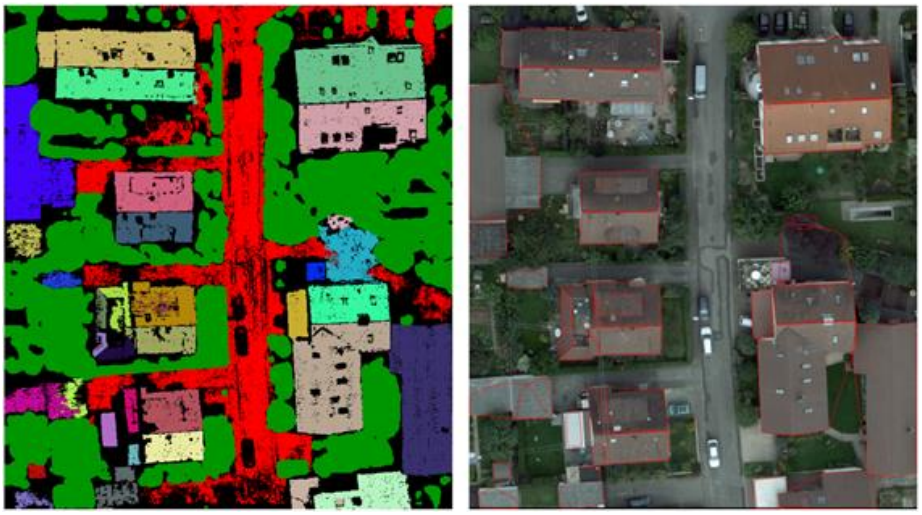
Ovaj proces obrade snimaka zemljišta fundamentalno se može podijeliti na klasifikaciju u kategorije, segmentaciju na osnovu kategorije i detekciju objekata od interesa[1]. Klasifikacija podrazumijeva razvrstavanje svakog dobijenog snimka u jednu ili više predefinisanih kategorija. U slučajevima klasifikacije aero-snimaka zemljišta, ovdje se radi o razvrstavanju cijelih slika ili njihovih pravougaonih podsegmenata u kategorije kao što su brdovit teren, naseljeno mjesto, rijeka, put i slično, kako je ilustrovano Slikom 1.1.

Slika 1.1 – Primjeri triju kategorija zemljišta za klasifikaciju (slijeva udesno: naseljeno područje, rijeka, luka)

Pored klasifikacije, bitan zadatak pri obradi aero-snimaka zemljišta je i segmentacija snimka u regione od interesa. Proces segmentacije obično podrazumijeva postojanje kategorija, slično kao pri klasifikaciji, ali se svaki snimak individualno dijeli na regione koji pripadaju pojedinačnim, predefinisanim, klasama, bilo na nivou piksela ili na nivou blokova. Uz procese

[1] Sve formalne definicije pojmova, objašnjenja i izvođenja, kao i neophodna teoretska osnova za razumijevanje predloženog rješenja, predstavljeni su u Glavi 2 i Glavi 3. Organizacija disertacije data je u poglavlju 1.2.

segmentacije i klasifikacije, često se koristi i detekcija objekata, čiji je zadatak određivanje pozicije i dimenzija objekata od interesa na dobijenim snimcima. U slučaju obrade aero-snimaka zemljišta, ovo bi moglo da podrazumijeva detekciju vozila, krovova, ulica i slično. Slika 1.2 prikazuje segmentisanu sliku i sliku na kojoj je izvedena detekcija objekata.



Slika 1.2 [2] *– Lijevo: segmentisan aero-snimak naseljenog mjesta, desno: detekcija krovova na aero-snimku naseljenog mjesta*

Manuelni proces razvrstavanja snimaka u kategorije, kao i ručno segmentisanje snimaka, naziva se označavanje (alternativno, labeliranje ili anotiranje). Ovo ručno označavanje predstavlja dugotrajan, vremenski i resursno zahtjevan, proces, te predstavlja značajan utrošak čovjek-sati pri procesu analize aero-snimaka zemljišta. Korištenjem pristupa za automatsku obradu, omogućava se automatsko označavanje zemljišta i na taj način olakšavaju prethodno navedeni zadaci planiranja, a budući da, zahvaljujući dostupnosti tehnologija za akviziciju, trenutno postoji velika količina podataka, vrlo često dovoljna za obučavanje, zadaci klasifikacije, segmentacije i detekcije mogu se veoma efikasno izvesti korištenjem odgovarajućih pristupa mašinskim učenjem, koji se, suštinski, oslanjaju na postojeće skupove označenih podataka kako bi se konstruisali i obučili autonomni sistemi koji su u stanju da generalizuju i automatizuju pomenute procese obrade. Ovaj vid automatizacije omogućava ekspertima iz polja da čovjek-sate ulažu u kompleksnije zadatke analize umjesto na označavanje dobijenih snimaka.

1.1 CILJEVI I DOPRINOS DISERTACIJE

Sam zadatak klasifikacije i segmentacije zemljišta predstavlja vrlo istražen problem za koji postoje brojna rješenja [3], no pristup projektovanju odgovarajuće arhitekture za njegovo izvođenje često je neizvediv bez odgovarajućeg istraživača ili usko specijalizovanog znanja na nivou istraživačkih ili eventualno razvojnih poslova. Pored projektovanja funkcionalnog rješenja sa visokom tačnošću klasifikacije i segmentacije, kakvo je dato u ovoj disertaciji, od suštinskog interesa pri projektovanju ovakvih sistema za različite upotrebe je koherentan pristup razvoju sa skupom pravila koja bi se mogla iskoristiti pri projektovanju sličnih specijalizovanih mreža, s ciljem brzog dolaska do rješenja i minimizacije utrošenih čovjek-sati pri razvoju. Na ovaj način, predloženo rješenje za automatsku klasifikaciju i segmentaciju aero-snimaka zemljišta, koje je glavni doprinos ovog rada, prošireno je odgovarajućim skupom pravila koji je doveo do njegovog efikasnog izvođenja i koji se može iskoristiti u sličnim primjenama za dobijanje sličnih rezultata. Razvoj pomenutog rješenja popraćen je i razvojem odgovarajućeg skupa pravila i smjernica za njegovo dobijanje, sa ciljem povećavanja upotrebne vrijednosti dobijenog proizvoda. Razvijanje ovog skupa zahtijevalo je dodatne eksperimente, ne direktno vezane za klasifikaciju i segmentaciju aero-snimaka, i analizu postojeće literature i dostupnosti postojećih pravila razvoja u oblasti projektovanja i razvoja vještačkih neuronskih mreža.

1.1.1 Klasifikacija, segmentacija i detekcija objekata na aero-snimcima zemljišta

U praksi, najčešće korišteni pristupi oslanjali su se na opis slika zemljišta na osnovu obilježja niskog nivoa, često semantički predstavljajući slike zemljišta kao teksture, za svrhu klasifikacije. Bilo da se radi o izvlačenju obilježja na osnovu boje, histogramom ili drugačije, na osnovu većeg broja kanala kod multispektralnih slika ili različitim predefinisanim deskriptorima niskog nivoa, čak i u slučajevima kada bi ove metode davale relativno dobre rezultate, njihova mana je obično bila u nedostatku više semantičke reprezentacije sadržaja ovih slika, odnosno u nedostatku korištenja obilježja visokog nivoa. Korištenjem konvolucionih neuronskih mreža i dubokog učenja, ove fundamentalne mane su prevaziđene, uz uvođenje dodatnih zahtjeva za izračunavanje i potrebe za akvizicijom velikih skupova podataka za obučavanje.

Uz adekvatne resurse i početne uslove, pristupi sa deskriptorima obilježja niskog nivoa, kao i pristupi sa neuronskim mrežama daju rezultate prema posljednjim standardima. Najefikasniji pristupi bazirani na deskriptorima obilježja niskog nivoa uključuju ekstrahovanje obilježja niskog nivoa predefinisanim algoritmima, postprocesiranje obilježja kao i statističke

klasifikatore. Glavna prednost ovakvih pristupa je mogućnost učenja obilježja na osnovu relativno malih skupova za obučavanje, posebno u slučajevima korištenja deskriptora niske dimenzionalnosti. S druge strane, duboke neuronske mreže, posebno konvolucione neuronske mreže, sekvencijalno kombinuju obilježja s ciljem dobijanja veoma duboke semantičke reprezentacije sadržaja, što se pokazuje kao najbolji pristup za ovu vrstu problema, posebno kada se koriste mreže predobučavane na sličnom skupu [4].

Za svrhu ovog istraživanja implementirana je neuronska mreža obučavana u dva koraka, uz predobučavanje, koja je korištena za rješavanje zadataka klasifikacije i segmentacije (odnosno klasifikacije regiona od interesa). Korišteni su UCMerced [4] i NWPU-RESISC45 [5] skupovi podataka, te su poređene performanse prethodnih najboljih implementacija sa predloženom. Dodatno, inspirisano prethodnim istraživanjima [6], mreža za detekciju i algoritam za segmentaciju i detekciju primijenjeni su na skupu podataka Geološkog Prikupljanja u SAD-u (eng. *U.S. Geological Survey*, USGS) [7].

Implementirana neuronska mreža sa svojim varijantama predstavlja primarni doprinos ove disertacije koji se može sumirati sljedećim stavkama.

- → Implementirana je duboka konvoluciona neuronska mreža koja rješava problem klasifikacije zemljišta na UCMerced skupu sa tačnošću od 98.61%, što je najbolji rezultat u odnosu na sve prethodne pristupe, u trenutku publikovanja dobijenih rezultata.
- → Predstavljen je, i pokazan kao efikasan, pristup sa modifikacijom postojeće arhitekture neuronske mreže, uz učenje sa transferom[2], za klasifikaciju aero-snimaka zemljišta. Ilustrovane su i pokazane slične upotrebe modifikacije postojećih arhitektura, kako bi se dodatno pokazala validnost pristupa.
- → Predstavljen je nov, dvofazni, pristup obučavanju neuronske mreže sa ciljem postizanja veće tačnosti klasifikacije i segmentacije uz dodatne ilustracije načina korištenja ovog pristupa.
- → Dat je pristup adaptaciji sistema za klasifikaciju za svrhu segmentacije, gdje je neuronska mreža obučena za zadatak klasifikacije iskorištena, uz algoritamski pristup prolaska kroz snimak klizećim prozorom, za segmentaciju ulazne slike proizvoljno visoke rezolucije.

[2] Učenje sa transferom, kao i svi ostali koncepti neophodni za razumijevanje, izloženi su i objašnjeni u ovoj disertaciji u glavama koja prethode izlaganju metodološkog pristupa i konačnog rješenja.

Korištenjem dobijenih pravila za razvoj specijalizovanih neuronskih mreža, implementiran je pristup klasifikaciji aero-snimaka zemljišta, čija je akvizicija izvedena sa autonomne letilice [8]. Implementirani publikovani pristup [1], kojim su postignuti najbolji rezultati u oblasti, ovdje je detaljno razrađen, sa svom metodologijom, pravilima i smjernicama, koje su dovele do konačnog rješenja i koje su izvedene u procesu razvoja proizvoda.

1.1.2 Metodologija razvoja neuronskih mreža za obradu aero-snimaka zemljišta

Posljednja decenija, uprkos ranijim predviđanjima i očekivanjima, okarakterisana je izrazitom i naglom ekspanzijom oblasti vještačke inteligencije i mašinskog učenja. Iako stvarni početak ekspanzije može da se datira ranije, realizacija konvolucionih neuronskih mreža za detekciju teksta u dokumentima implementirana u vidu LeNet arhitekture [9] često se smatra početkom ove ekspanzije. Ova realizacija pokazala je upotrebljivost neuronskih mreža za stvarne slučajeve upotrebe i dala početni impuls oblasti dubokog učenja i dodatni impuls cijelog oblasti mašinskog učenja.

Od samog početka decenije pa do danas, oblasti vještačke inteligencije i mašinskog učenja u rapidnom su rastu, čega je indikator i godišnji AiIndex [10] izvještaj koji ukazuje na nekoliko značajnih rastućih trendova. Broj objavljenih radova u ovoj oblasti, broj konferencija, GitHub projekata, kompanija u oblasti mašinskog učenja, kao i rezultati brojnih sprovedenih anketa nedvojbeno ukazuju na ovu činjenicu. Mašinsko učenje, u nekom obliku, postaje integralni dio razvojnog mehanizma mnogih komercijalnih proizvoda sa značajnim i gotovo uniformnim udjelom u većini oblasti nauke i inžinjerstva, od agrikulture do aeronautike. Kako sistemi vještačke inteligencije bivaju sve više ugrađivani u komercijalne sisteme, to se sve više pojavljuje potreba za njihovom standardizacijom, pojačavanjem sigurnosti i formalizacijom zakona s obzirom na etičke implikacije.

Težina ove oblasti i njenih proizvoda postala je jasna već 2015. godine kada je pokazano da su sistemi vještačke inteligencije namijenjeni za klasifikaciju slika u stanju da klasifikuju slike sa tačnošću koja prevazilazi prosječnu tačnost u slučaju manuelne klasifikacije. ImageNet [11] baza slika, koja sadrži milione primjeraka označenih fotografija, savladana je sa greškom od 4.94% na top-5 metrici [12], što je, u to vrijeme, bila neočekivana novost koja je omogućila objelodanjenje cijele oblasti široj publici. Nakon ovoga, usljedila je serija implementacija neuronskih mreža koje su bile u stanju da prevaziđu ljude u različitim specijalizovanim oblastima, uključujući, između ostalog i kao najznačajnije, savladavanje kineske igre Go [13],

prevazilaženje tačnosti dermatologa u detekciji karcinoma kože [14], prevođenje teksta sa kineskog na engleski jezik sa ljudskom tačnošću [15], pobjeđivanje tima ljudi u strateškoj računarskoj igri Dota 2 [16] i prevazilaženje ljudske tačnosti u detekciji karcinoma prostate (sa detekcijom ograničenom na snimke, bez istorije i anamneze) [17].

Ovi rezultati pokazuju enorman potencijal vještačkih neuronskih mreža, kao glavnog mehanizma i predmeta istraživanja oblasti mašinskog učenja, za primjenu u specijalizovanim oblastima. Činjenica je da pomenute publikacije predstavljaju vrhunac u oblasti mašinskog učenja i da su izrazito zavisne od dostupnih resursa autora. S druge strane, mnogi relativno manje značajni projekti koriste slične ili iste pristupe za realizaciju bliskih problema ili problema iz bliskih oblasti. Zbog toga se javlja potreba za metodologijom projektovanja i razvoja arhitektura vještačkih neuronskih mreža za specifične oblasti. Autori pomenutih publikacija nedvojbeno posjeduju izuzetan nivo tehničkog znanja, vještine i instrumentalnih resursa za razvoj ovakvih mreža, no često, kako se pokazuje iz literature, projektovanje specijalizovanih arhitektura zahtijeva značajno vrijeme provedeno u eksperimentisanju, istraživanju i testiranju različitih, potencijalno i često pogrešnih, pristupa i često se ovi resursi ulažu u isto rješenje sa više različitih strana. Za komercijalne svrhe i projekte koji su djelomično povezani, od značaja bi bio skup metoda i pristupa, koji bi specifikovao pravila i smjernice za lakši razvoj arhitektura neuronskih mreža za specifične, dijelom istražene, namjene. Ovakva metodologija bi omogućila inžinjerima koji nisu nužno stručnjaci u oblasti mašinskog učenja da razvijaju vještačke neuronske mreže za komercijalne ili interne svrhe u relativno kratkom vremenskom periodu, bez potrebe za detaljnim udubljivanjem u oblast mašinskog učenja s ciljem povećanja dostupnosti vještačke inteligencije različitim oblastima inžinjerskog i naučnog djelovanja, što je pretpostavljeni cilj razvoja velikog broja softverskih biblioteka za rad sa neuronskim mrežama.

S obzirom na to da je konstrukcija sveobuhvatne metodologije razvoja neuronskih mreža izuzetno obiman zadatak, smisleno je fokusirati napore na konstrukciju skupa pravila za rješavanje jednog segmenta relativno poznatih i istraženih specijalizovanih oblasti. Oblast klasifikacije, bilo da se radi o aero-snimcima ili nekoj drugoj oblasti, pokazuje se kao pogodna za formulisanje pravila i smjernica, jer su pristupi klasifikaciji i segmentaciji često vrlo slični, bez obzira na autore.

U oblasti klasifikacije i segmentacije aero-snimaka zemljišta, kao i u većini oblasti gdje je klasifikacija elementaran dio problema, ne postoji definisana metodologija, te je od suštinskog interesa njegovo izvođenje, dato ovdje kao osnova za dalji razvoj.

Prema tome, kao sekundarni cilj i doprinos, ova disertacija sadrži metodološka pravila za razvoj neuronskih mreža specijalizovanih za svrhu klasifikacije i segmentacije snimaka zemljišta, kao i preporuke za adaptaciju i modifikaciju postojećih arhitektura konvolucionih neuronskih mreža za slične primjene. Doprinos u vidu metodologije razvoja može da se sumira u sljedeće stavke.

- → Predstavljen je jedinstven pregled oblasti mašinskog učenja i razvoja konvolucionih neuronskih mreža za svrhu klasifikacije, segmentacije i detekcije sa postojećim modelima mreža i njihovim arhitekturama potrebnim za dobijanje upotrebljivih rezultata pri obradi aero-snimaka zemljišta.
- → Pored teoretskog pregleda i obrazloženja, predstavljeni su i postojeći modeli neuronskih mreža kao i pristupi obučavanju, uz objašnjenje njihove interne funkcionalnosti, neophodni za razumijevanje predstavljenog meta-algoritma.
- → Date su modifikovane varijante izvođenja jednačina neuronskih mreža i neuronskih jedinica, predstavljene na način razumljiv razvojnom inžinjeru van oblasti, sa ciljem olakšavanja razumijevanja postojećih modela neuronskih mreža i kao minimalna neophodna osnova za razumijevanje i obrazloženje datog meta-algoritma.
- → Pored rješenja za klasifikaciju i segmentaciju aero-snimaka zemljišta, dati su primjeri, na izvedenim eksperimentima iz vlastitih radova vezanih za ovu disertaciju, koji ilustruju funkcionalnost svih tehnika korištenih pri izradi dobijenog rješenja.
- → Glavni doprinos u razvoju metodologije projektovanja i razvoja neuronskih mreža dat je kao meta-algoritam koji, na osnovu predstavljenih arhitektura i obrazloženja njihove funkcionalnosti, omogućava modifikaciju postojećih ili implementaciju novih neuronskih mreža za svrhe klasifikacije i segmentacije. Ovaj meta-algoritam izveden je na osnovu detaljne analize postojeće literature i eksperimenata izvedenih za svrhe ove disertacije, opisanih u sklopu disertacije.

1.2 ORGANIZACIJA DISERTACIJE

Glavni problem koji ova disertacija rješava je problem automatske klasifikacije i segmentacije zemljišta sa ciljem dobijanja efikasnog sistema koji može da zamijeni manuelni rad, odnosno ručno označavanje zemljišta sa aero-snimaka. Dodatno, kako je već rečeno, radi povećavanja upotrebljivosti rješenja u drugim oblastima i omogućavanja proširenja istog modela i pristupa na druge klasifikcione oblasti, uz izvedeno rješenje ovdje je prezentovan i metodološki proces koji je do njega doveo, zajedno sa pravilima i smjernicama izvedenim iz različitih studija (i onih izvedenih u istraživanju za svrhe ove disertacije i onih publikovanih od strane drugih autora).

Kako bi se dobijeno rješenje i odgovarajući metodološki pristupi detaljno i sistematično obrazložili, neophodno je izložiti sve teoretske i praktične pristupe korištene pri izvođenju metodološkog pristupa, prije izlaganja samog rješenja. Na ovaj način, jedan od ciljeva ovog rada, predstavljanje skupa pravila upotrebljivih inžinjerima van oblasti mašinskog učenja i klasifikacije aero-snimaka, može biti adekvatno izložen, bez značajne potrebe za dodatnom literaturom. Uz ovu glavu, koja predstavlja uvod u temu disertacije, i zaključak dat u Glavi 7, ostatak disertacije organizovan je na sljedeći način.

Glava 2 opisuje matematički model vještačke neuronske mreže od perceptrona do višeslojnih neuronskih mreža sa različitim tipovima slojeva. Tu su objašnjeni matematički koncepti uvođenja nelinearnosti kroz aktivacione funkcije, funkcije gubitka, kao i procedura optimizacije sa različitim pristupima optimizaciji. Ova glava predstavlja teoretski uvod i pregled i njen glavni doprinos je u adekvatnoj sistematizaciji osnovnih koncepata iz oblasti mašinskog učenja s ciljem kasnijeg referenciranja u glavama koje razrađuju implementaciju rješenja i metodološki pristup koji je do nje doveo. Ovdje je dato i sopstveno matematičko izvođenje propagacije unazad sa ciljem ilustracije funkcionalnosti neuronske mreže i pokazivanja matematičke prirode slojeva, odnosno matematičkog modelovanja procesa optimizacije. Izloženi su svi neophodni koncepti za teoretsko razumijevanje prezentovanog metodološkog pristupa razvoju specijalizovanih mreža za klasifikaciju i segmentaciju.

U Glavi 3 predstavljen je programski model, odnosno implementaciono proširenje matematičkog modela koje uključuje detalje o pristupu programskoj realizaciji matematičkog modela vještačke neuronske mreže, te pristupima i procedurama obučavanja i ugradnje neuronskih mreža s obzirom na oblast primjene, ograničenja hardvera, kao i ostala ograničenja koja su posljedica implementacije u realnom okruženju. Ovdje su izloženi neki specifični

problemi koji se pojavljuju pri razvoju neuronskih mreža, kao i praktična rješenja koja se mogu iskoristiti za njihovo rješavanje. Glava 3 ne sadrži apstraktna metodološka pravila, ali sadrži načine rješavanja praktičnih problema, uključujući sprečavanje overfitinga, podešavanje performansi, numeričku stabilnost i paralelizaciju. Ovo su praktični problemi koji se neminovno pojavljuju pri projektovanju i obučavanju, te ih je neophodno obrazložiti, kako bi se prezentovani pristup mogao bazirati na apstraktnijim koracima rješavanja, podrazumijevajući da su osnovni implementacioni problemi prevaziđeni.

Glava 4 daje pregled oblasti primjene sa specifičnim arhitekturama za svaku od oblasti kao i objašnjenjima funkcionalnosti arhitektura ili dijelova arhitektura mreža najčešće korištenih u specifičnim oblastima. Budući da se metodološki pristup koji je doveo do implementacije rješenja za klasifikaciju i segmentaciju aero-snimaka zemljišta zasniva na adaptaciji postojećih mreža i njihovih modula, ova glava je od suštinskog značaja za materijal izložen u Glavi 5, jer metodološki pristup zahtijeva poznavanje nekoliko važnih arhitektura neuronskih mreža koje se najčešće koriste za rješavanje poznatih problema. Glava 4 daje pregled i objašnjenje načina rada ovih mreža (odnosno, njihovih individualnih modula) kao kompleksniji element neophodan za formulaciju apstraktnog metodološkog pristupa.

Pozivajući se na prezentovane arhitekture iz Glave 4, individualne arhitekturalne probleme iz Glave 3 i njihova rješenja, kao i matematičku apstrakciju neuronske mreže date u Glavi 2, Glava 5 daje osnovna metodološka pravila za projektovanje novih ili specijalizaciju postojećih arhitektura s obzirom na oblast klasifikacije i segmentacije, sa ciljem ekspanzije glavnog doprinosa ovog rada iz oblasti klasifikacije i segmentacije aero-snimaka zemljišta na oblast klasifikacije i segmentacije slike uopšteno, za primjenu u drugim specijalizovanim oblastima.

Konačno, u Glavi 6 predstavljeno je implementirano rješenje za klasifikaciju i segmentaciju aero-snimaka zemljišta, bazirano na prezentovanom skupu metodoloških pravila. Ovo poglavlje izlaže glavni doprinos ovog rada, odnosno specijalizovanu neuronsku mrežu za rješavanje konkretnog problema klasifikacije i segmentacije aero-snimaka u svrhu zamjene manuelnog rada, gdje je isti model neuronske mreže koji je iskorišten za klasifikaciju, bez dodatnog obučavanja, iskorišten u svrhu segmentacije korištenjem algoritamske augmentacije. Projektovanje i implementacija modela pratili su izložena pravila i služe kao validacija njihove funkcionalnosti.

1.3 PUBLIKACIJE

Neuronska mreža za klasifikaciju aero-snimaka, sa predloženim dvofaznim pristupom, objavljena je, kao dio rezultata istraživanja za ovu disertaciju, u [1]. Predloženi pristup baziran je na modifikaciji postojeće arhitekture, pa je modifikacija uključena u metodologiju razvoja, a jedan od primjera na osnovu kojih je taj dio metodologije izveden objavljen je u [18], sa dvogranom mrežom za klasifikaciju multispektralnih slika.

Pristupi razvoju novih mreža na osnovu razumijevanja domena izvedeni su na osnovu pomenutih radova i postojeće literature i dodatno ilustrovani u [19], uz pristup augmentaciji ulaznih podataka koji je dodatno proširen ovdje. Objavljeni rad rješavao je isti problem koji je riješen algoritamskim pristupom i publikovan u [20] i [21], kao dio projekta *G5355 "Biological Method (Bees) for Explosive Detection"*, a ovdje korišten za izvođenje metodologije razvoja neuronskih mreža.

Prikazani pristupi ansamblovskoj (eng. *ensemble*) klasifikaciji, usko vezani za višegrane modifikacije mreža, objavljeni su u [22], a poređenje algoritamskog pristupa sa modifikacijom uz minimalne izmjene dato je u [23].

Predstavljeni dio metodologije koji se odnosi na ekvivalenciju projektovanja algoritma sa projektovanjem arhitekture neuronske mreže inspirisan je rudimentarnom ekvivalencijom sa filtriranjem po obilježjima niskog nivoa, datom u [24].

Dodatne publikacije korištene za derivaciju metodologije i kao podrška predloženom rješenju uključuju i [25] i [26].

2 MATEMATIČKI MODEL NEURONSKE MREŽE

2.1 PREGLED

Kao osnova za konstrukciju metodologije razvoja klasifikacionih neuronskih mreža koja će u ovom radu biti korišćena za razvoj mreže za klasifikaciju i segmentaciju aero-snimaka, u ovoj glavi je teorijski predstavljen i detaljno razrađen matematički model vještačke neuronske mreže, sa svim neophodnim pojmovima i konceptima za konstrukciju prezentovanog rješenja.

Poglavlje 2.2 sadrži osnovne jednačine, većinom poznate istraživačima i razvojnim inžinjerima u oblasti mašinskog učenja, uključujući jednačine konvolucionih neuronskih mreža, kao i izvođenje algoritma propagacije unazad za obučavanje neuronskih mreža. Predstavljen je jedan osnovni model sa ciljem jasnijeg predočavanja ekvivalencije programskog modela neuronske mreže, često predstavljenim blokovima slojeva, sa njenim matematičkim modelom, odnosno, jednačinom koja je predstavlja, te ekvivalencije procesa obučavanja sa optimizacijom funkcije gubitka.

Poglavlja 2.3, 2.4 i 2.5 sadrže najčešće korištene aktivacione funkcije, funkcije gubitka i optimizatore upotrebljavane pri razvoju najkorištenijih modela neuronskih mreža. U ovim poglavljima date su preporuke za korištenje individualnih aktivacionih funkcija i optimizatora, njihove prednosti i mane, kao i najčešće korištene numeričke vrijednosti parametara optimizatora.

Poglavlje 2.6 predstavlja neke od osnovnih problema koji se mogu pojaviti pri obučavanju neuronskih mreža, te daje uvod u potencijalna rješenja.

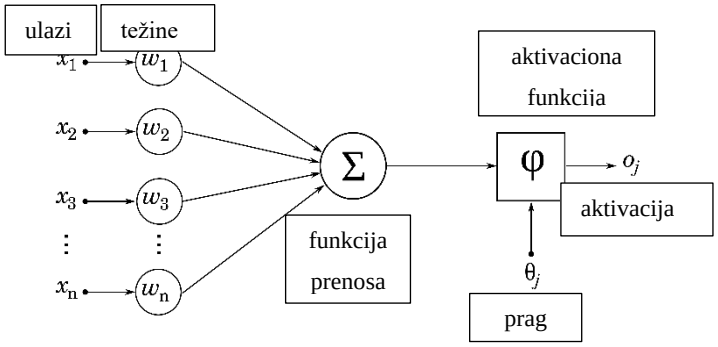
Poglavlje 2.7 predstavlja veliki broj postojećih arhitektura neuronskih mreža, upotrebljivih za praktične zadatke, sa preporukama za njihovo korištenje, te izdvajanjem ključnih i najčešće korištenih arhitektura, kao i nekih njihovih značajnih osobina.

2.2 MODEL VJEŠTAČKOG NEURONA I NEURONSKE MREŽE

Na prvi pogled, s obzirom na naziv, moglo bi se pogrešno zaključiti da vještačke neuronske mreže predstavljaju matematički model stvarnih neuronskih mreža, no to nije slučaj. U nekoj mjeri, vještački neuroni su inspirisani načinom funkcionisanja bioloških neurona, ali njihov matematički model predstavlja značajno uprošćenje rada stvarnih neurona i jedina stvarna relacija između biološkog neurona i vještačkog neurona je u tome da se i jedni i drugi

međusobno povezuju sa ciljem učenja i realizacije neke funkcionalnosti. Biološki neuroni svoju funkcionalnost realizuju putem impulsa, dok, iako postoje pulsne neuronske mreže [27], vještački neuroni predstavljaju samo matematički model elementarnog konstituenta veće arhitekture koja je u stanju da bude obučena za rješavanje nekog specifičnog problema i koja se, opet po konvenciji, naziva neuronska mreža, pa je zbog toga alternativni naziv za vještački neuron novoformirani termin *perceptron*.

Bez obzira na naziv, vještački neuroni, u klasičnim mrežama (odnosno, nerekurentnim mrežama gdje individualni neuroni ne posjeduju memoriju), modeluju se kao spoj jedinice za sumiranje ulaza (kod jednostavnih neuronskih mreža: transfer funkcija ili prenosna funkcija) i jedinice za delinearizaciju (najčešže nazvane: aktivaciona funkcija).



Slika 2.1 [28] – *Model perceptrona*

Prema tome i sa Slike 2.1, vještački neuron predstavlja jednostavan klasifikator koji prihvata vektor ulaza, na slici označen kao $\boldsymbol{x}$, vektor težina koje predstavljaju sinaptičke težine za vještački neuron, na slici označen kao $\boldsymbol{w}$ i nad njihovim skalarnim proizvodom primjenjuje aktivacionu funkciju, tipično neku nelinearnu funkciju. U rijetkim slučajevima kada se koristi jedan neuron, aktivaciona funkcija je najčešće Hevisajdova funkcija, kako bi se omogućila binarizacija izlaza, odnosno klasifikacija u dvije klase. Prema tome, matematički model jednog neurona, kada je u pitanju model sa slike, može da se predstavi jednačinom (2.1).

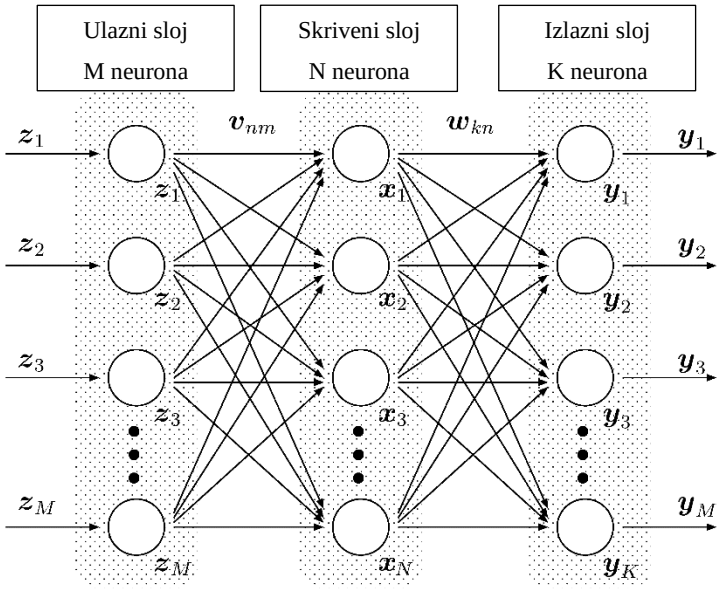
$$y = \varphi(\boldsymbol{x} \cdot \boldsymbol{w}) \tag{2.1}$$

Jedan neuron predstavlja elementarni klasifikator koji operiše nad skupom podataka fiksne dimenzionalnosti iz vektorskog prostora dimenzionalnosti vektora $\boldsymbol{x}$ i klasifikuje na broj klasa određen kvantizacijom skalarnog izlaza y. Tipična upotreba jednog neurona je u obliku binarnog klasifikatora, gdje se jedna klasa označava izlaznom vrijednošću 0 ili -1, a druga izlaznom vrijednošću 1. Jednoneuronski klasifikator koduje informacije o razlikama između

klasa unutar vektora težina $\boldsymbol{w}$. Tako se svaki element ulaznog vektora $\boldsymbol{x}$ skalira linearno odgovarajućim elementom vektora težina $\boldsymbol{w}$.

Vrijednosti elemenata vektora težina, odnosno težine, mogu da se podešavaju ručno ili na osnovu eksperimenata ili na osnovu propagacije gradijenata u postupku koji se naziva nadgledano učenje.

Jedan neuron ne predstavlja značajan model klasifikatora, ali njegova se upotrebna vrijednost i klasifikaciona moć povećavaju kada se oni povezuju u arhitekturu koja se naziva vještačka neuronska mreža. U najjednostavnijem obliku, vještačka neuronska mreža predstavlja niz slojeva vještačkih neurona, gdje se svakom neuronu u svakom sloju na ulaze prosljeđuju izlazi neurona prethodnog sloja, pri čemu je sloj definisan kao niz nezavisnih vještačkih neurona. Slika 2.2 ilustruje način organizacije više neurona u jednostavnu vještačku neuronsku mrežu koja se zove višeslojna vještačka neuronska mreža ili, ustaljeno ali teoretski neispravno zbog neopravdanosti rekurzivnog definisanja perceptrona, višeslojni perceptron.



Slika 2.2 [29] – *Troslojna neuronska mreža*

U slučaju višeslojne neuronske mreže, često se koriste troslojne mreže, gdje se slojevi redom nazivaju ulazni, skriveni i izlazni sloj. Naravno, ne postoji suštinsko ograničenje za broj skrivenih slojeva mreže i postoje implementacije kod kojih je broj međusobno povezanih slojeva daleko veći [30]. Višeslojne neuronske mreže sa potpuno povezanim slojevima (kako je predstavljeno Slikom 2.2) najčešće sadrže jedan ili dva skrivena sloja, jer se pokazuje da, u slučaju ovakve arhitekture, dodatni skriveni slojevi ne daju značajno bolje rezultate [31]. Osnovni razlog za dodavanje više slojeva je obezbjeđivanje nelinernosti klasifikatora, odnosno

mogućnosti nelinearne separacije različitih klasa u domenu ulaznih podataka. Tipično, u ovakvim arhitekturama, skriveni slojevi sadrže veći broj neurona od ulaznih i izlaznih slojeva.

Ukoliko se posmatraju dva susjedna sloja u višeslojnoj neuronskoj mreži, može se primijetiti da je svaki neuron iz jednog sloja povezan sa svakim neuronom iz sljedećeg, pa se tako vektori težina za neurone sljedećeg sloja mogu spojiti u matricu. Prema tome, propagacija signala od jednog sloja ka sljedećem može da se opiše jednačinom (2.2), gdje indeksi i označavaju broj sloja na kom se parametar nalazi.

$$\boldsymbol{y_{i+1}} = \varphi_i(\boldsymbol{W}_i \boldsymbol{y}_i) \tag{2.2}$$

Važno je uočiti da je izlaz iz svakog sloja vektor. Za svaka dva susjedna sloja postoji odgovarajuća matrica težina. Matrica $\boldsymbol{W}_i$ je kvadratna samo u slučaju da dva susjedna sloja sadrže isti broj neurona. U opštem slučaju, matrica $\boldsymbol{W}_i$ nije kvadratna i njen broj redova i kolona odgovara dimenzionalnosti ulaznog i izlaznog vektora, odnosno broju neurona i-tog i $(i+1)$-tog sloja. Takođe, moguće je da svaki sloj, pa i svaki neuron svakog sloja, ima drugačiju aktivacionu funkciju.

Koncept slojeva i neurona postoji radi uprošćenja i vizuelizacije klasifikatora, ali, fundamentalno i matematički, višeslojna neuronska mreža može da se predstavi kao serija matematičkih operacija (funkcija) koje preslikavaju vektor iz domena ulaznih podataka (podataka koji se klasifikuju) u domen labela (klasa prema kojima se vrši razvrstavanje ulaznih podataka).

Radi uopštenja ovog osnovnog modela, potrebno je naglasiti da se često na izlaze funkcije prenosa individualnih neurona u svakom od slojeva dodaju novi parametri koji omogućavaju pravljenje ofseta na izlazu funkcije prenosa radi preferiranja izlaza određenih neurona iz prethodnog sloja. Ovaj član se često naziva prednaponski član (eng. *bias*) i prošireni model za dva susjedna sloja je, na osnovu ovoga, dat jednačinom (2.3).

$$\boldsymbol{y}_{i+1} = \varphi_i(\boldsymbol{W}_i \boldsymbol{y}_i + \boldsymbol{b}_i) \tag{2.3}$$

Očigledno, dimenzionalnost prednaponskog člana mora da bude jednaka dimenzionalnosti izlaznog vektora. Prednaponski član se dodaje prije primjene aktivacione funkcije, kako bi ulaz aktivacione funkcije bio u odgovarajućem opsegu, i kako bi se izbjeglo zasićenje usljed nelinearnosti aktivacione funkcije.

2.2.1 Konvolucione neuronske mreže

Konvolucione neuronske mreže predstavljaju proširenje modela višeslojnih neuronskih mreža dodavanjem nove vrste sloja, odnosno matematičke operacije, u jednačinu koja predstavlja neuronsku mrežu.

Naime, u praktičnim primjenama, ulaz neuronske mreže, odnosno podatak za klasifikaciju, je često slika, a budući da se radi o digitalnom domenu, takav podatak predstavlja se kao dvodimenziona matrica (najčešće nazvanom dvodimenzionim tenzorom, iako se ne radi o matematičkom konceptu tenzora) intenziteta piksela, u slučaju jednokanalne slike, ili kao višedimenziona matrica (tenzor), u slučaju višekanalne slike. Kako je ulaz slika, organizacija ulaznog podatka koduje dio informacija o ulaznom skupu (konkretno, činjenicu da se susjedni pikseli u ekranskom prostoru čuvaju na susjednim elementima ulaznog tenzora). Drugim riječima i uopštenije, odabir organizacije podataka u elemente tenzora dovedenog na ulaz mreže formira pristrasnost mreže ka rješavanju optimizacionog problema na neki specifičan način. U slučaju klasične višeslojne neuronske mreže, ta semantika se gubi, jer primijenjene operacije množenja vektora matricom ni na koji način ne razlikuju redoslijed podataka na ulazu za vrijeme faze obučavanja mreže. Drugim riječima, pikseli ulazne slike mogli su da budu (konzistentno na svim elementima ulaznog skupa, sa istom permutacijom ulaza) preuređeni na bilo koji način bez uticaja na tačnost klasifikatora. S druge strane, ukoliko se konstruiše operacija čija je semantika takva da se uzima u obzir poredak ulaznih podataka, mreža kao cjelina dobija mogućnost da jednostavnije stekne razumijevanje o ulaznom skupu podataka, jer je semantika tog skupa ugrađena u operacije koje se izvode. U tipičnim primjenama, operacija matrične konvolucije sa klizećim prozorom koristi se u onim slučajevima kada se očekuje da postoji semantička veza između susjednih piksela u ulaznoj slici i kada operacija konvolucije nad takvom slikom daje jednako semantički povezan rezultat. U daljem tekstu, termin *semantika susjednosti piksela* je korišten za označavanje ove veze. Ovdje je riječ o semantici koja je smislena ljudskom posmatraču, dok se u eksperimentima za svrhu ove disertacije pokazalo da to ne mora da bude slučaj, odnosno, moguće je koristiti operaciju konvolucije i onda kada ulaz nije slika sa čovjeku smislenom semantikom susjednosti.

Kodovanje semantike ulaznog skupa izmjenom jednačine mreže fundamentalna je tehnika pri projektovanju arhitektura neuronskih mreža, i čini bazu metodologije projektovanja i razvoja koja je ovdje prezentovana. Ekvivalencijom arhitekture neuronske mreže (njenih parametara i hiperparametara) sa jednačinom koja je opisuje izvedeni su neki od koraka meta-algoritma za projektovanje predstavljenog u Glavi 5. Iako izvođenje konkretnih jednačina za

neku specifičnu arhitekturu neuronske mreže (ovdje, konvolucione neuronske mreže), nije neophodno za razumijevanje predstavljenog skupa pravila, ovdje je dato kao studija slučaja i ilustracija pomenute ekvivalencije na primjeru dodavanja nove operacije koja koduje semantiku susjednosti piksela.

Konkretno, jedna takva operacija je operacija matrične konvolucije, odnosno konvolucije fiksnim matričnim kernelom. Konvolucija predstavljena jednačinom (2.4) predstavlja dvodimenzionu diskretnu konvoluciju.

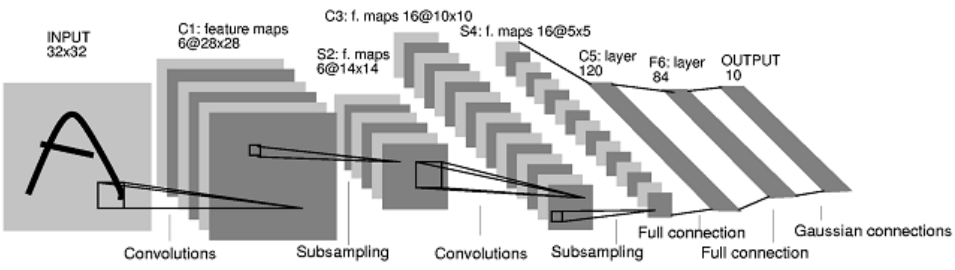
$$f[n_x, n_y] * g[n_x, n_x] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i,j] \cdot g[n_x - i, n_y - j] \tag{2.4}$$

Sličan pristup funkcionisao bi i u slučaju da se radi o jednodimenzionom ulaznom podatku gdje susjedne vrijednosti ulaznog vektora odgovaraju stvarnoj povezanosti njima odgovarajućih podataka. Ovakav slučaj je, na primjer, sa odmjercima audio zapisa ili, opštije, sekvencijalnim mjerenjima vrijednosti neke fizičke veličine. U tom slučaju, jednačina (2.4) bi se mogla zapisati u jednodimenzionom obliku, datom jednačinom (2.5).

$$f[n_x] * g[n_x] = \sum_{i=-\infty}^{\infty} f[i] \cdot g[n_x - i] \tag{2.5}$$

U praktičnom slučaju sa konvolucionim neuronskim mrežama, dimenzionalnost ulaznog podatka je konstantna, pa se konvolucija izvodi samo na semplovanom dijelu domena, odnosno samo nad pikselima ulazne slike. Drugim riječima, diskretna konvolucija je ograničena u slučajevima upotrebe kod neuronskih mreža, kao rezultat činjenice da su ulazne slike ograničenog broja piksela, odnosno ograničenih dimenzija.

Prema tome, konvoluciona mreža je svaka mreža koja sadrži bar jedan konvolucioni sloj, gdje je konvolucioni sloj skup operacija koji izvršava neku varijantu konvolucije nad izlazima prethodnog sloja ili nad ulaznim podacima.



Slika 2.3 – Arhitektura LeNet [9] *konvolucione mreže, slika iz originalnog rada*

Generalno, i kako se vidi sa Slike 2.3, konvolucione neuronske mreže ne moraju nužno da sadrže samo konvolucione slojeve i najčešće sadrže, pored ostalih slojeva, slojeve za agregaciju i potpuno povezane slojeve, odnosno klasične slojeve višeslojnih neuronskih mreža[3].

Intuitivno, u slučaju jednokanalne dvodimenzione slike, rezultat konvolucije konvolucionim kernelom nad nekim ulazom predstavlja matricu čiji elementi označavaju postojanje obilježja definisanog kernelom na ulazu. Na primjer, u slučaju da kernel predstavlja obilježje kose linije, pri konvoluciji će na izlazu biti proizvedene numerički veće vrijednosti za one slučajeve kada je klizeći prozor konvolucije prešao preko kosih linija na ulazu, efektivno kodujući na izlazu lokacije sa ulaza gdje je postojala kosa linija.

U slučaju da je ulazna slika višekanalna, tada se konvolucija vrši trodimenzionim kernelom, pri čemu je jednačine potrebno proširiti za ovu dimenzionalnost. Ovako se i ulazni podaci i konvolucioni kerneli matematički tretiraju kao tenzori, jer su oni pogodno uopštenje s obzirom na proizvoljnu dimenzionalnost prostorno strukturisanih ulaznih podataka. Naravno, postoje varijacije po pitanju izvođenja operacije konvolucije, te nije standardizovan način na koji se konvolucija mora izvoditi kako bi se sloj zvao konvolucionim. U ovom poglavlju je predstavljen samo najčešći vid implementacije konvolucije kod konvolucionih neuronskih mreža. Kako je ranije rečeno, konvoluciona mreža je bilo koja mreža čija jednačina sadrži operaciju konvolucije.

Slično kao i u slučaju jednog neurona, sličnost načina funkcionisanja konvolucionih slojeva sa načinom funkcionisanja bioloških neurona je gotovo nepostojeća i koristi se samo iz tradicionalnih razloga i čuvanja ustanovljene nomenklature. Suštinski gledano, parametrizacija operacije konvolucije kod neuronskih mreža događa se na nivou konvolucionog kernela, slično kako se parametrizacija individualnog neurona odvijala na nivou vektora težina, ili kako se parametrizacija sloja odvijala na nivou matrice težina.

Dobijanje vrijednosti parametara operatora koji čine mrežu je cilj optimizacije i njihova promjena i podešavanje u cilju poboljšavanja tačnosti klasifikacije u definisanom domenu, predstavlja obučavanje klasifikatora za definisani zadatak.

U slučaju konvolucionih slojeva, kako je rečeno, parametrizabilni članovi su konvolucioni kerneli (odnosno numeričke vrijednosti elemenata kernela). Samim tim, nameće

[3] O različitim slojevima i njihovim upotrebama će biti više riječi u Glavi 3.

se problem broja kernela (nerijetko se umjesto termina kernel, u oblasti mašinskog učenja, koristi termin filtar, odnosno, u ovom slučaju, broj filtara). Jedan konvolucioni sloj, po definiciji [26], može da sadrži više konvolucionih kernela (odnosno, više filtara), te na taj način omogući detekciju većeg broja različitih obilježja na ulaznoj slici ili ulaznom tenzoru. U tom slučaju, za svaki od filtara, sloj će da generiše po jedan izlaz. Ovi izlazi nazivaju se mapama obilježja (eng. *feature maps*), pa se tako za svaki konvolucioni sloj kaže da sadrži mape obilježja, odnosno neki broj mapa obilježja. Brojevi mapa obilježja u slojevima, kao i brojevi slojeva različitog tipa, često se nazivaju *hiperparametrima* mreže. Izbor slojeva i njihovih parametara i njihovo međusobno povezivanje predstavljaju arhitekturu neuronske mreže.

Ukoliko je dat sloj izlaza dimenzija $n \times n$ i ukoliko se koristi $m \times m$ kernel $\boldsymbol{\omega}$, rezultujuća mapa obilježja (bez agregacije) će biti veličine $(n - m + 1) \times (n - m + 1)$. Kako bi se izračunao ulaz nekog piksela y_{ij}^{l} u l-tom sloju, neophodno je sumirati sve doprinose (skalirane težinama komponenata konvolucionog kernela), s obzirom na kernel, iz elemenata prethodnog sloja, kako je dato jednačinom (2.6).

$$y_{ij}^{l} = \varphi\left(\sum_{a=0}^{m-1}\sum_{b=0}^{m-1} \omega_{ab} y_{(i+a)(j+b)}^{l-1}\right) \tag{2.6}$$

Jednačina ima dati oblik ukoliko se radi o jednokanalnim slojevima, ali ukoliko postoji više kanala u prethodnom sloju, odnosno više mapa obilježja, i više konvolucionih kernela u trenutnom sloju, tada je tenzor kernela četvorodimenzion. Tako, ukoliko je k indeks mape obilježja (odnosno konvolucionog kernela) u l-tom konvolucionom sloju, a c_{l-1} broj kanala u prethodnom sloju, jednačina (2.6) može se generalizovati na jednačinu (2.7).

$$y_{ijk}^{l} = \varphi\left(\sum_{a=0}^{m-1}\sum_{b=0}^{m-1}\sum_{c=0}^{c_{l-1}-1} \omega_{abck} y_{(i+a)(j+b)c}^{l-1}\right) \tag{2.7}$$

Ovo je jedan od načina implementacije konvolucionog sloja. Suštinski gledano, operacija konvolucije omogućava mreži da koduje podatak o susjednosti piksela koji se na ulazu nalaze na susjednim mjestima. U literaturi se pod nazivom *konvolucija*, u kontekstu neuronskih mreža, koriste i različite varijacije na operaciju predstavljenu jednačinama (2.5) i (2.6).

Jedan alternativni način implementacije konvolucionog sloja je odvojeno iteriranje dvodimenzionim kernelom kroz sve mape obilježja prethodnog sloja. U ovom slučaju, kerneli

su dvodimenzioni, pa je rezultujući broj mapa obilježja jednak proizvodu broja mapa obilježja u prethodnom sloju i broja kernela u trenutnom sloju. Jednačina (2.8) daje ovu verziju konvolucionog sloja. Zavisno od primjene i cilja, konvolucioni slojevi mogu da se adaptiraju.

$$y^{l}_{ij(kc_{l-1}+c)} = \varphi\left(\sum_{a=0}^{m-1}\sum_{b=0}^{m-1} \omega_{abk} y^{l-1}_{(i+a)(j+b)c}\right), c \in [\![0, c_{l-1}]\!] \tag{2.8}$$

Pri konstrukciji konvolucionih mreža, najčešće se naizmjenično povezuju konvolucioni slojevi i slojevi za agregaciju. Striktno matematički gledano, ovo nije neophodno, ali se uvođenjem slojeva za agregaciju smanjuje broj parametara koje je potrebno optimizovati, te se omogućava učenje invarijantnosti na rotaciju i skaliranje [32]. Agregaciju je moguće raditi na više različitih načina od kojih su najkorišteniji agregacija po maksimalnoj vrijednosti (eng. *max pooling*) i agregacija po prosječnoj vrijednosti (eng. *average pooling*), gdje se kroz ulazni tenzor iterira prozorom i iz prozora na izlaz šalje maksimalna, odnosno prosječna vrijednost piksela, respektivno. Za agregaciju koja se radi na nivou kanala, za svrhu smanjivanja veličine izlaza, nerijetko se koristi izraz pododmjeravanje (eng. *subsampling*), odnosno sloj za agregaciju ima ulogu sloja za pododmjeravanje.

Potpuno povezani slojevi ne koduju podatke o susjednosti ulaza, pa redoslijed dovođenja podataka na ulaz ovih slojeva nije bitan, dok god se svi podaci dovode konzistentno pri svakom izračunavanju. Zbog toga je moguće izlaz bilo kog konvolucionog sloja (i bilo kog sloja uopšte) dovesti na ulaz potpuno povezanog sloja. Matematički gledano, potrebno je preurediti izlazni tenzor u vektor. Ova operacija se često naziva izravnavanjem (eng. *flatten*), pa se i odgovarajući sloj naziva slojem za izravnavanje. Sloj za izravnavanje ne sadrži parametre koji se optimizuju, ali se naziva slojem jer sadrži operacije koje semantički ne pripadaju ni jednom drugom sloju.

Ove četiri vrste slojeva (konvolucioni sloj, sloj za agregaciju, sloj za izravnavanje i potpuno povezan sloj) predstavljaju sve tipove slojeva potrebne da se konstruiše veliki broj jednostavnih konvolucionih mreža za različite namjene. Na primjer, originalna LeNet konvoluciona mreža sadržavala je samo ove tipove slojeva.

2.2.2 Obučavanje nad labeliranim skupom i nadgledano učenje

U praksi, skup oznaka, odnosno labela, često je konstruisan za domen od interesa za koji se radi obučavanje, od strane stručnjaka iz oblasti, koji u procesu obučavanja ima ulogu *labelara* čiji je zadatak formiranje uređenih parova elemenata iz ulaznog skupa i njima

odgovarajućih elemenata iz skupa labela, odnosno oznaka. Budući da termin *označen podatak* u matematici i softverskom inžinjerstvu nosi drugačiju semantiku, ovdje je korišten termin labela, odnosno labeliranje, za vektor iz označenog skupa i označavanje.

Na osnovu prethodno izloženog, jasno je na koji način, uz adekvatno odabrane parametre (težine veza i kernela) neki klasifikator može da posjeduje odgovarajuću klasifikacionu moć da preslikava vektore iz domena ulaznih podataka u domen labela, no i dalje ostaje problem pronalaženja vrijednosti ovih parametara. Kako rastu broj slojeva, veličine kernela i broj kernela po sloju, tako raste i broj parametara koje je potrebno odabrati, odnosno optimizovati.

Zbog ovoga koristi se pristup koji se naziva nadgledano učenje. Ovaj pristup podrazumijeva da postoji već definisan i labeliran skup podataka za obučavanje. Tačnije, postoji skup podataka koji sadrži ulazne slike različitih klasa (u slučaju klasifikacije) i odgovarajuće oznake klasa (labele). Uopštenije, skup za obučavanje sadrži primjere ispravnog mapiranja iz domena podataka u domen labela, odnosno iz domena podataka u ciljni domen.

Pod pretpostavkom da neka proizvoljna neuronska mreža sadrži idealno optimizovane parametre, kada se na ulaz prvog sloja mreže dovede neki podatak (u opštem slučaju tenzor), odnosno kada se matematička funkcija koja predstavlja mrežu izračuna sa tim podatkom kao argumentom, dobija se izlaz koji odgovara labeli tog podatka. U praktičnim slučajevima, sa dobro optimizovanim parametrima, mreža će na izlazu da daje vektor najsličniji labeli očekivanog podatka (gdje se sličnost određuje nekom odabranom funkcijom).

S druge strane, ako se parametri mreže odaberu nasumično, tada, bez obzira na ulaz mreže, na izlazu mreže može da se očekuje šum. Činjenica da je mreža modelovana kao diferencijabilna matematička funkcija garantuje da se nad slobodnim parametrima može vršiti optimizacija. Drugim riječima, iako mreža inicijalizovana nasumično daje šum, postojanje labeliranih parova u skupu za obučavanje omogućava da se za svaki poznat ulaz pronađe greška mreže[4] na izlazu (budući da postoji definisana ispravna i očekivana vrijednost na izlazu). Ova greška se potom može unazad propagirati kroz mrežu sa ciljem korekcije vrijednosti parametara koje su do nje dovele. Jednačina mreže mora da bude diferencijabilna kako bi ova analiza bila moguća, pa se, prema tome, za jednačinu mreže ne može odabrati proizvoljna matematička funkcija.

[4] Termin *greška* biće preciznije definisan u nastavku (poglavlje 2.2.3). Ovdje se može smatrati da je greška neka definisana razlika između dobijenog i očekivanog vektora.

Formalno, skup za obučavanje predstavlja skup uređenih parova vektora iz prostora ulaznih podataka i iz prostora labela, odnosno skup uređenih parova ulaznih i njihovih očekivanih izlaznih vektora.

Postupak izvršavanja funkcije mreže, odnosno dobijanja izlaza na osnovu dovedenog ulaza naziva se propagacija unaprijed ili inferencija. Termin propagacija unazad odnosi se na propagaciju greške od izlaza mreže do početnog sloja i ključni je mehanizam za optimizaciju parametara mreže na osnovu skupa za obučavanje. Skup parametara mreže često se označava sa θ, a skup svih ulaza sa X.

2.2.3 Propagacija unazad

Kako je ranije pomenuto, parametri mreže ažuriraju se na osnovu ustanovljene greške na izlazu. Funkcija kojom se izračunava greška na izlazu naziva se funkcija greške ili funkcija gubitka (eng. *loss function*) i može biti proizvoljno definisana. Najčešće, kod višeslojnih neuronskih mreža sa potpuno povezanim slojevima, često nazivanih mrežama sa propagacijom unaprijed (eng. *feed forward neural networks*), kao funkcija gubitka koristi se funkcija srednjekvadratne greške, kako je dato jednačinom (2.9) (ovdje je odabran koeficijent $\frac{1}{2}$ radi dobijanja pogodnijeg oblika prvog izvoda, *N* predstavlja ukupan broj elemenata vektora, a $\boldsymbol{x}$ posmatrani ulaz mreže iz skupa *X*).

$$E(\boldsymbol{x}, \theta) = \frac{1}{2N} \sum_{i=0}^{N-1} (\hat{y}_i - y_i)^2 \tag{2.9}$$

Minimizacija funkcije greške vrši optimizaciju nad skupom za obučavanje i skupom parametara, pri čemu je $\hat{y}_i$ dobijeni izlaz mreže, a y_i očekivani izlaz mreže na i-tom elementu vektora, za svaki element iz skupa za obučavanje. Minimizacija se vrši izračunavanjem, za svaku težinu w_{ij}^k, gradijenta funkcije greške $\frac{\partial E}{\partial w_{ij}^k}$. Kako se jednačina (2.8) može razviti po članovima sume, gradijenti se mogu izračunavati individualno za svaki izlaz, a sumirati na kraju (jednačina (2.10)).

$$\frac{\partial E(\boldsymbol{x}, \theta)}{\partial w_{ij}^k} = \frac{1}{N} \sum_{d=0}^{N-1} \frac{\partial}{\partial w_{ij}^k} \left(\frac{1}{2} (\hat{y}_d - y_d)^2 \right) = \frac{1}{N} \sum_{d=0}^{N-1} \frac{\partial E_d}{\partial w_{ij}^k} \tag{2.10}$$

Budući da je cijela mreža serija ugniježdenih funkcija, izračunavanje gradijenata na nekom sloju na osnovu gradijenata u sljedećem sloju može da se dobije primjenom lančanog

pravila. Neka je a_j^k izlaz funkcije prenosa, odnosno vrijednost prije primjene aktivacione funkcije na j-tom neuronu k-tog sloja i neka je o_j^k izlaz j-tog neurona u k-tom sloju. Po definiciji, $\delta_j^k \equiv \frac{\partial E}{\partial a_j^k}$ naziva se greškom j-tog neurona u k-tom sloju, i predstavlja gradijent koji će biti propagiran unazad. Na osnovu ovoga, jednačine (2.11)-(2.13) predstavljaju izračunavanje greške na slojevima, gdje je r_k broj neurona u k-tom sloju.

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k}\frac{\partial a_j^k}{\partial w_{ij}^k} \tag{2.11}$$

$$\frac{\partial a_j^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_{ij}^k}\left(\sum_{l=0}^{r_{k-1}} w_{lj}^k o_l^{k-1}\right) = o_i^{k-1} \tag{2.12}$$

$$\frac{\partial E}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} \tag{2.13}$$

Na osnovu definicije a_l^{k+1} date jednačinom (2.14) i definicije δ_l^{k+1} date jednačinom (2.15), dobijaju se jednačine za propagaciju unazad zapisane jednačinama (2.16)-(2.18).

$$a_l^{k+1} = \sum_{j=0}^{r_k} w_{jl}^{k+1}\varphi\left(a_j^k\right) \tag{2.14}$$

$$\delta_j^k = \sum_{l=0}^{r_{k+1}} \delta_l^{k+1}\frac{\partial a_l^{k+1}}{\partial a_j^k} \tag{2.15}$$

$$\frac{\partial a_l^{k+1}}{\partial a_j^k} = w_{jl}^{k+1}\varphi'\left(a_j^k\right) \tag{2.16}$$

$$\delta_j^l = \sum_{1=0}^{r_{k+1}} \delta_l^{k+1} w_{jl}^{k+1}\varphi'\left(a_j^k\right) = \varphi'\left(a_j^k\right)\sum_{l=0}^{r_{k+1}} w_{jl}^{k+1}\delta_l^{k+1} \tag{2.17}$$

$$\frac{\partial E}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} = \varphi'\left(a_j^k\right)o_i^{k-1}\sum_{l=0}^{r_{k+1}} w_{jl}^{k+1}\delta_l^{k+1} \tag{2.18}$$

Jednačina (2.16) se nekad naziva i jednačina propagacije unazad, jer opisuje način izračunavanja gradijenata greške u trenutnom sloju na osnovu gradijenata i izlaza na sljedećem

sloju. Ova jednačina daje vektore grešaka koje se često nazivaju delta vektorima. Na osnovu ovih jednačina, odnosno na osnovu dobijenih gradijenata, mogu da se ažuriraju parametri mreže izvođenjem propagacije unazad nad svim parovima iz skupa za obučavanje [26] [33] [34] [35]. Ovo ažuriranje dato je jednačinom (2.19).

$$\Delta w_{ij}^{k} = -\alpha \frac{\partial E(\boldsymbol{x}, \theta)}{\partial w_{ij}^{k}} \tag{2.19}$$

Dakle, promjena vrijednosti težina određuje se na osnovu dobijenog gradijenta, skalirano sa koeficijentom učenja (eng. *learning rate*) α. Glavni razlog za postojanje koeficijenta učenja je to što korekcija težina treba da bude proporcionalna gradijentu, ali ne nužno njemu jednaka. Dodatno, iz praktičnih razloga, obučavanje se ne vrši nad cijelim skupom za obučavanje, već iterativno nad podskupovima iz skupa za obučavanje koji se nazivaju grupe (eng. *batch*), ili se kroz cijeli skup za obučavanje prolazi više puta u prolazima koji se nazivaju epohe, pa se koeficijent učenja koristi kako bi se regulisao doprinos ažuriranju parametara u svakoj iteraciji ili epohi.

Bitno je naglasiti da se sličan pristup koristi bez obzira na tip sloja, jer su različiti tipovi slojeva ekvivalentni različitim matematičkim funkcijama, pa je postupak izvođenja generalno veoma sličan.

U slučaju konvolucionih neuronskih mreža čiji su posljednji slojevi potpuno povezani slojevi, matematički gledano, gradijenti propagirani unazad od izlaza do ulaza prvog potpuno povezanog sloja mogu da se propagiraju dalje kroz konvolucione slojeve, pri čemu se ažuriranje gradijenata radi u skladu sa operacijom konvolucije.

Ukoliko se posmatra neki jednokanalni konvolucioni sloj, gdje je E funkcija greške i izračunate su vrijednosti greške za svaki izlaz, poznato je da su greške koje je potrebno izračunati parcijalni izvodi funkcije greške po svakom izlazu [26] [34] [36] [37], odnosno $\frac{\partial E}{\partial y_{ij}^{l}}$. Poznavajući greške dobijene sa sljedećeg sloja propagacijom unazad, mogu da se izračunaju gradijenti za svaku težinu primjenom lančanog pravila, pri čemu je neophodno sumirati doprinos svih izraza u kojima se promjenljiva pojavljuje, jer se u slučaju konvolucije i klizećeg kernela, gdje se klizeći kernel posmatra kao neuron, isti ulazi neurona aktiviraju više puta u toku jedne konvolucije.

$$\frac{\partial E}{\partial \omega_{ab}} = \sum_{i=0}^{n-m} \sum_{j=0}^{n-m} \frac{\partial E}{\partial x_{ij}^{l}} \frac{\partial x_{ij}^{l}}{\partial \omega_{ab}} = \sum_{i=0}^{n-m} \sum_{j=0}^{n-m} \frac{\partial E}{\partial x_{ij}^{l}} y_{(i+a)(j+b)}^{l-1} \tag{2.20}$$

Jednačina (2.20) daje način izračunavanja gradijenta u slučaju jednokanalne konvolucije, odnosno dvodimenzionog kernela i slijedi iz jednačine (2.4) i prethodno pokazanog postupka. U ovom slučaju, kako je rečeno, neophodno je izvršiti sumiranje po svim x_{ij}^{l} izrazima u kojima se ω_{ab} pojavljuje. Prema jednačinama propagacije unaprijed, poznato je da je $\frac{\partial x_{ij}^{l}}{\partial \omega_{ab}} = y_{(i+a)(j+b)}^{l-1}$. Kako bi se izračunao gradijent, potrebno je poznavati vrijednosti greške $\frac{\partial E}{\partial x_{ij}^{l}}$, nekad nazivanim deltama. Ove delta vrijednosti se relativno jednostavno izračunavaju, ponovo korištenjem lančanog pravila, kako se vidi iz jednačine (2.21).

$$\frac{\partial E}{\partial x_{ij}^{l}} = \frac{\partial E}{\partial y_{ij}^{l}} \frac{\partial y_{ij}^{l}}{\partial x_{ij}^{l}} = \frac{\partial E}{\partial y_{ij}^{l}} \frac{\partial}{\partial x_{ij}^{l}} \left(\varphi\left(x_{ij}^{l}\right)\right) = \frac{\partial E}{\partial y_{ij}^{l}} \varphi'\left(x_{ij}^{l}\right) \tag{2.21}$$

Kako je greška na trenutnom sloju već poznata $\frac{\partial E}{\partial y_{ij}^{l}}$, moguće je izračunati delte $\frac{\partial E}{\partial x_{ij}^{l}}$ na trenutnom sloju samo koristeći izvod aktivacione funkcije $\varphi'(x)$, a kako su poznate greške na trenutnom sloju, može se izračunati i gradijent s obzirom na težine koje koristi dati konvolucioni sloj.

Pored računanja promjena težina kernela u konvolucionom sloju, neophodno je propagirati greške nazad na prethodni (gledano od prvog konvolucionog sloja) sloj. Ponovo, korištenjem lančanog pravila, može da se dobije odgovarajuća jednačina (2.22).

$$\frac{\partial E}{\partial y_{ij}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{l}} \frac{\partial x_{(i-a)(j-b)}^{l}}{\partial y_{ij}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{l}} \omega_{ab} \tag{2.22}$$

Posmatrajući jednačine propagacije unaprijed, može se primijetiti da je $\frac{\partial x_{(i-a)(j-b)}^{l}}{\partial y_{ij}^{l-1}} = \omega_{ab}$. Ovo slijedi iz izraza za greške propagirane iz prethodnog sloja (počevši od posljednjeg konvolucionog sloja). Jedan od praktičnih problema koji se javljaju u slučaju propagacije unazad (on postoji i kod propagacije unaprijed, ali je ovdje obrađen detaljnije), odnosno izračunavanja gradijenata pri konvoluciji je problem ivica, odnosno popunjavanja ivičnih vrijednosti. On se najčešće rješava dodavanjem nula na pozicije koje bi se inače odmjeravale van slike, ponavljanjem ivičnih vrijednosti ili reflektovanjem ulaza po osi ruba.

Navedene jednačine za propagaciju unazad predstavljaju jednačine za slučaj gdje je konvolucija uprošćena na jednokanalne ulazne slike, odnosno na jednu mapu obilježja na ulazu. Zbog toga je matrica veza, ω, dvodimenziona. Jednačine se mogu uopštiti kako bi se u obzir uzeo i slučaj sa više kanala. Bitno je primijetiti da je konvolucija po različitim kanalima nezavisna, pa neki parcijalni članovi ne postoje u jednačinama, jer greška za dati kanal ne zavisi od ostalih kanala. Na osnovu toga i jednačine (2.4), mogu se izvesti uopštene jednačine (2.23)-(2.25).

$$\frac{\partial E}{\partial \omega_{abcd}} = \sum_{i=0}^{n-m} \sum_{j=0}^{n-m} \frac{\partial E}{\partial x_{ijd}^{l}} y_{(i+a)(j+b)c}^{l-1} \tag{2.23}$$

$$\frac{\partial E}{\partial x_{ijk}^{l}} = \frac{\partial E}{\partial y_{ijk}^{l}} \sigma'\left(x_{ijk}^{l}\right) \tag{2.24}$$

$$\frac{\partial E}{\partial y_{ijk}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \sum_{c=0}^{n_l-1} \frac{\partial E}{\partial x_{(i-a)(j-b)c}^{l}} \omega_{abkc} \tag{2.25}$$

Date jednačine predstavljaju jedan način implementacije konvolucionih slojeva i propagacije gradijenata kroz konvolucione slojeve. Naravno, moguće je konstruisati konvolucioni sloj kod kog bi postojala zavisnost između kanala ili gdje bi zavisnost bila drugačije definisana [38] [39]. Suštinski, ove jednačine su date kao pregled osnovnog koncepta propagacije gradijenata u slučaju slojeva koji nisu potpuno povezani.

Propagacija gradijenata kod slojeva za izravnavanje je trivijalna, budući da se radi samo o promjeni matematičke reprezentacije i odgovarajućih indeksa, a kako je sloj za izravnavanje bijektivan, propagacija gradijenata kroz njega je inverzna operacija operaciji izvedenoj pri propagaciji unaprijed, na istom sloju.

U slučaju slojeva za agregaciju, gradijenti se propagiraju u skladu sa vrstom agregacije. U slučaju pododmjeravanja po maksimumu, gradijenti se propagiraju samo na onaj element koji je pri pododmjeravanju imao maksimalnu vrijednost. Ostali elementi se ne mijenjaju i gradijenti se na njih ne propagiraju. Nakon ažuriranja težina, u nekoj narednoj iteraciji optimizacije, za isti sloj za agregaciju neki drugi element može dati maksimalnu vrijednost, pa se u toj iteraciji gradijenti propagiraju na njega. U slučaju pododmjeravanja po prosječnoj vrijednosti, gradijenti se propagiraju skalirani sa vrijednostima izlaza, odnosno inverzno operaciji usrednjavanja.

Propagacija gradijenata unazad nije jedini način na koji se parametri mreže mogu optimizovati, ali je ubjedljivo najkorišteniji pristup (u stohastičkoj varijanti [40] [41]) i njegov pronalazak je predstavljao jedan od preduslova za ubrzan rast oblasti mašinskog učenja.

Iako su jednačine konvolucionih mreža sa izvođenjem propagacije unazad poznati koncepti, ovdje su dati kao referenca uz specifične probleme koji se u praksi moraju riješiti (uključujući razrješavanje rubnih uslova pri konvoluciji, kao i različite vrste konvolucije) pri projektovanju i obučavanju neuronskih mreža. Dato je vlastito izvođenje propagacije unazad, relativno slično postupcima često izlaganim u literaturi, predstavljeno na minimalan način, kako bi se dala osnova za razumijevanje načina funkcionisanja optimizatora, te olakšao odabir optimizatora pri upotrebi predloženog skupa pravila datog u Glavi 5. Na osnovu izvedenih jednačina implementiran je i vlastiti alat za konstrukciju i obučavanje neuronskih mreža korišten u [22] i ovdje je dat radi potpunosti i ustanovljavanja terminologije.

2.3 AKTIVACIONE FUNKCIJE

Kako je ranije pomenuto, individualni vještački neuroni sadrže čvor za aktivaciju, odnosno primjenjuju delinearizaciju na rezultat sumiranja, odnosno izlaz funkcije prenosa. Primarna uloga aktivacionih funkcija u vještačkim neuronima je delinearizacija izlaza neurona, kako bi se individualni neuroni mogli koristiti za ulazne podatke čije klase nisu nužno linearno separabilne. Drugi razlog za korištenje aktivacionih funkcija je ograničavanje kodomena (odnosno domena izlaza), na neki zatvoren ili otvoren interval. Ovo omogućava bolju kontrolu nad vrijednostima pri inferenciji, kao i nad gradijentima pri propagaciji unazad.

Štaviše, određeni procesi se ne mogu modelovati bez delinearizacije, jer fundamentalno nisu linearni, jer kako bi se modelovao nelinearan proces, jednačina koja ga modeluje mora da bude nelinearna. Jednačina koja opisuje neuronsku mrežu bez grananja kao cjelinu biće linearna ukoliko u mreži ne postoji nelinearna aktivaciona funkcija. Uvođenjem nelinearnih elemenata u mrežu, odgovarajuća jednačina sadrži nelinearne članove i samim tim može, sa odgovarajućim odabirom koeficijenata, da modeluje daleko kompleksnije procese, odnosno, daleko kompleksnija preslikavanja sa ulaznog prostora na izlazni (odnosno, na prostor labela).

Iako, u principu, bilo koja funkcija može da bude aktivaciona funkcija, obično se za aktivacione funkcije biraju one funkcije čiji oblik delinearizuje ulaz na traženi način i čiji izvod je pogodan za izračunavanje pri propagaciji unazad. Ovdje su navedene osnovne i najčešće korištene aktivacione funkcije sa njihovim slučajevima upotrebe.

Linearna aktivaciona funkcija je najjednostavnija funkcija za aktivaciju, i, suštinski, predstavlja skaliranje izlaza nekim konstantnim koeficijentom. Jednačina (2.26) opisuje linearnu aktivacionu funkciju, a jednačina (2.27) njen izvod.

$$R(x,k) = kx \tag{2.26}$$

$$R'(x,k) = k \tag{2.27}$$

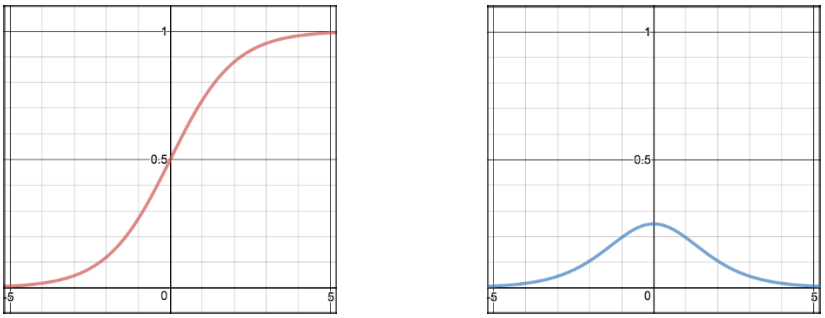
Ova funkcija se relativno rijetko koristi, jer, pored činjenice da je linearna, vrijednost njenog izvoda ne zavisi od njegovog ulaza, pa je, samim tim, njena primjena vrlo ograničena. Takođe, ova funkcija može dovesti do eksplozije gradijenata.

Sigmoidna aktivaciona funkcija (ili logistička funkcija) i njen izvod dati su jednačinama (2.28) i (2.29). Ova funkcija preslikava ulaz na interval $[0, 1]$. Pored toga, ova funkcija je monotona, neprekidna i kontinualno diferencijabilna, što su često vrlo povoljne osobine za aktivacionu funkciju. Vrlo često se koristi u slučajevima kada je ulaz potrebno delinearizovati u neki pozitivan opseg. Slika 2.4 daje grafike sigmoidne funkcije i njenog izvoda.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.28}$$

$$S'(x) = S(x) \cdot (1 - S(x)) \tag{2.29}$$

Ova funkcija je vrlo povoljna za korištenje pri klasifikaciji i često je aktivaciona funkcija za posljednji sloj mreže. S druge strane, ova funkcija može da izazove problem nestajućih gradijenata, pogotovo ukoliko je ulaz značajno različit od nule. Sigmoidne funkcije često saturišu gradijente, odnosno zaustavljaju efektivnu propagaciju unazad. Budući da izlaz nije centriran oko nule, korištenje ove funkcije u skrivenim slojevima može dovesti do pomjeranja izlaza i lošije tačnosti mreže [42] [43] [44] [45].



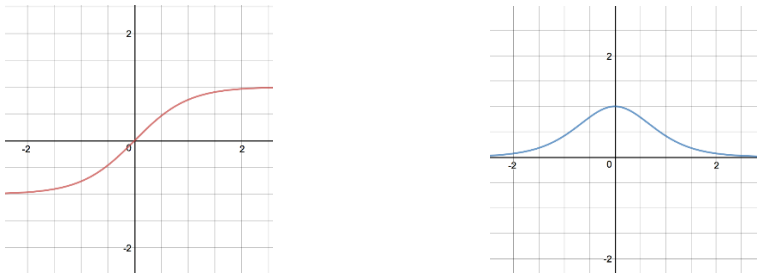
Slika 2.4 [46] – *Sigmoidna aktivaciona funkcija (lijevo) i njen izvod (desno)*

Vrlo slična funkcija sigmoidnoj je **hiperbolni tangens**. Ova aktivaciona funkcija i njen izvod dati su jednačinama (2.30) i (2.31), dok su odgovarajući grafici dati na Slici 2.5. Za razliku od sigmoidne funkcije, hiperbolni tangens preslikava ulaz na interval $[-1, 1]$, centriran oko nule. Ova funkcija je nelinearna, i zbog centriranosti oko nule obično se preferira u odnosu na sigmoidnu. Njene matematičke osobine slične su osobinama sigmoidne funkcije.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.30}$$

$$tanh'(x) = 1 - tanh(x)^2 \tag{2.31}$$

Gradijent je strmiji za ovu funkciju u odnosu na sigmoidnu, ali i dalje postoji problem nestajućeg gradijenta.



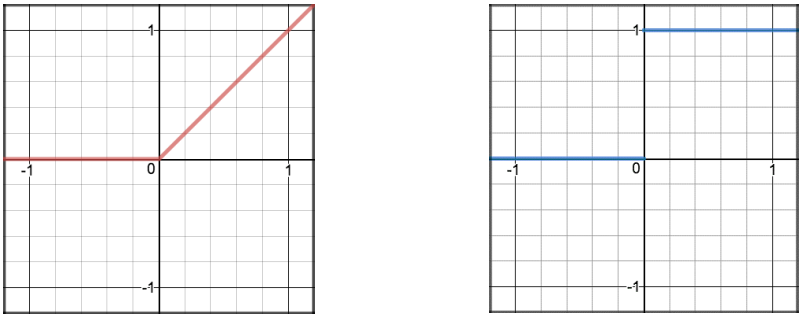
Slika 2.5 – Hiperbolni tangens i njegov izvod

Ispravljačka funkcija (eng. *Rectified Linear Unit*), poznatija kao **ReLU**, izrazito je jednostavna funkcija, ali nije linearna i obično daje bolje performanse od sigmoidne funkcije. Ova funkcija i njen izvod dati su jednačinama (2.32) i (2.33), a odgovarajući grafici Slikom 2.6.

$$R(x) = \max(x, 0) \tag{2.32}$$

$$R'(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2.33}$$

Ova funkcija uspješno izbjegava problem nestajućeg gradijenta i manje je izračunski zahtjevna od sigmoidne funkcije i hiperbolnog tangensa. Generalno, koristi se samo u skrivenim slojevima, jer rijetko ima smisla na samom izlazu neuronske mreže. Glavni nedostatak ove aktivacione funkcije je problem takozvanog „umirućeg gradijenta“ ili „problem umiruće ReLU“, gdje se može desiti takvo ažuriranje težine da se dati neuron više ne aktivira. Pored toga, ova funkcija, zbog činjenice da preslikava ulaz na neograničen interval može da izazove ekploziju aktivacije [47].



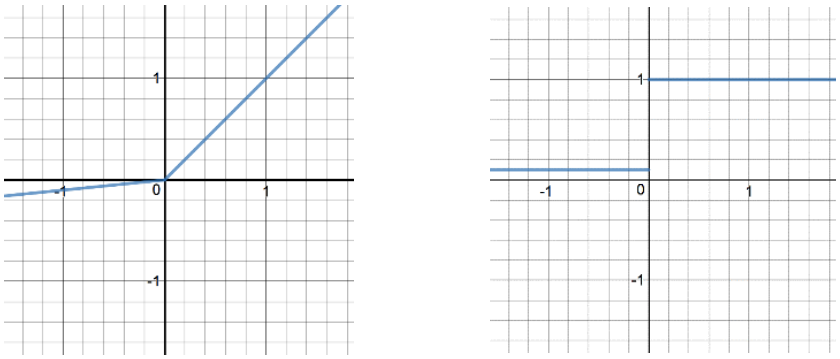
Slika 2.6 – ReLU i njen izvod

Cureća ReLU (eng. *LeakyReLU*) je varijanta ReLU funkcije, gdje se umjesto konstante za vrijednosti ulaza manje od nula koristi linearna funkcija konstantnog nagiba. Jednačine (2.34) i (2.35) opisuju ovu funkciju i njen izvod, a Slika 2.7 daje odgovarajuće grafike.

$$R(x,\alpha) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{2.34}$$

$$R'(x,\alpha) = \begin{cases} 1, & x > 0 \\ \alpha, & x \leq 0 \end{cases} \tag{2.35}$$

Ova funkcija se pokazala kao ključna za prevazilaženje ljudske tačnosti u klasifikaciji nad ImageNet skupom [12]. Koeficijent kosine α je obično vrlo mala vrijednost (reda veličine 0.01), što rješava problem umirućeg gradijenta. S druge strane, ova aktivaciona funkcija se nekad pokazuje kao inferiorna u odnosu na sigmoidnu funkciju i hiperbolni tangens. Pored ovoga, postoji i parametarska ReLU, obično nazvana **PReLU**, funkcija kod koje se parametar α uči kao parametar mreže.



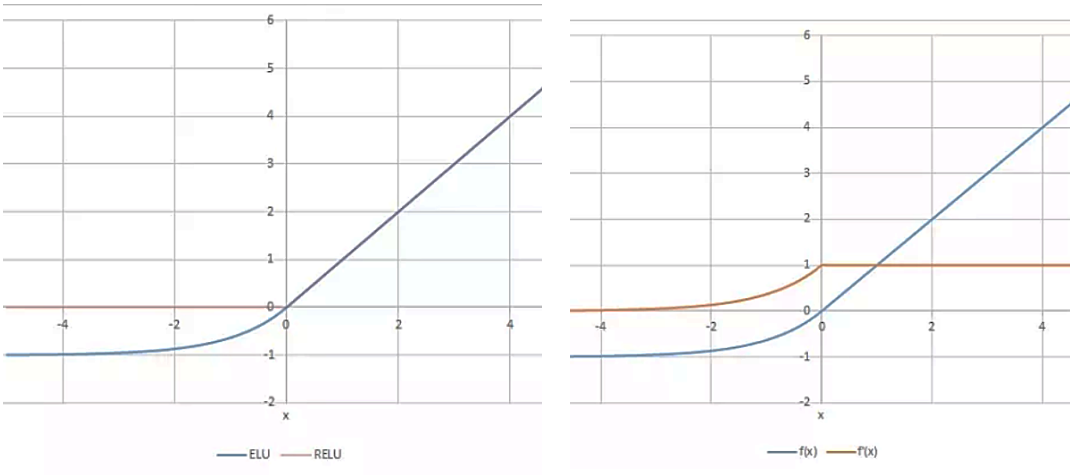
Slika 2.7 – Cureća ReLU i njen izvod

Eksponencijalna linearna jedinica ili ELU (eng. *Exponential Linear Unit*) slična je curećoj ReLU funkciji, s tom razlikom da se za negativne vrijednosti ponaša kao eksponencijalna funkcija. Jednačine (2.36) i (2.37) opisuju ovu funkciju i njen izvod, a Slika 2.8 sadrži odgovarajuće grafike.

$$R(x,\alpha) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases} \tag{2.36}$$

$$R'(x,\alpha) = \begin{cases} 1, & x > 0 \\ \alpha e^x, & x \leq 0 \end{cases} \tag{2.37}$$

Ova funkcija je glatka i asimptotski se približava $-\alpha$, kako negativan ulaz postaje sve manji. Glavna mana ekponencijalne linearne jedinice je njen potencijal da izazove eksploziju aktivacije, jer preslikava ulaz na otvoren interval.



Slika 2.8 – Eksponencijalna linearna jedinica (plavim) i ReLU (lijevo) i eksponencijalna linearna jedinica (plavim) i njen izvod (desno)

Pored navedenih funkcija, često se koristi i **softmax** funkcija. S obzirom na svoje specifičnosti, ova funkcija se najčešće koristi na izlaznom sloju neuronske mreže. U opštem slučaju, softmax funkcija je generalizacija sigmoidne funkcije koja preslikava vektor iz nekog vektorskog prostora u vektor iz vektorskog prostora niže dimenzionalnosti. U slučaju neuronskih mreža, kao aktivaciona funkcija, softmax funkcija prihvata dva argumenta: vektor i indeks nekog njegovog elementa; i na osnovu njih izračunava vrijednost. Ova funkcija, pored potencijalne redukcije dimenzionalnosti, ima normalizacionu ulogu. Vrijedi napomenuti da se ova funkcija može koristiti u različitim varijantama. Jednačina (2.38) daje opšti oblik softmax funkcije, za j-ti član K-dimenzionog izlaznog vektora i ulazni vektor $\boldsymbol{z}$.

$$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.38}$$

Pored pomenutih aktivacionih funkcija, moguće je, principijelno, koristiti bilo koju funkciju kao aktivacionu funkciju. Naravno, ovo nije često slučaj. Postoje i druge funkcije koje se ponekad koriste u svrhu aktivacije: apsolutna vrijednost, stepena funkcija, eksponencijalna

funkcija, logaritamska funkcija, Hevisajdova funkcija i parametrizovana linearna funkcija (koeficijent pravca se uči kao parametar mreže).

Izbor aktivacione funkcije suštinski zavisi od konkretne primjene, ali u većini slučajeva, pri konstrukciji arhitekture neuronske mreže, za skrivene slojeve koristi se ReLU ili neka slična varijanta, a za izlazne sigmoidna funkcija, hiperbolni tangens ili softmax.

2.4 FUNKCIJE GUBITKA

Funkcija gubitka (eng. *loss function*), nekad nazvana i funkcija koštanja (eng. *cost function*) ili funkcija greške (eng. *error function*) je funkcija čija vrijednost se minimizira za vrijeme procesa obučavanja i njome se, u svakom koraku obučavanja, određuje greška na izlazu mreže koja će se koristiti u procesu propagacije unazad s ciljem podešavanja težina. Dakle, vrijednost funkcije gubitka se izračunava na osnovu dobijenog i očekivanog izlaza za svaki neuron. Ova funkcija, u principu, može da bude bilo koja matematička funkcija koja na neki način određuje relaciju između dva ulazna podatka, ali ona se najčešće bira tako da njen rezultat predstavlja vrijednost koju je potrebno minimizovati.

U procesu propagacije unazad, cilj optimizacije treba da bude da za svaki par ulaz-izlaz iz skupa za obučavanje funkcija gubitka daje minimalnu vrijednost. Drugim riječima, optimizator koji se primjenjuje na mreži cilja da minimizira vrijednost funkcije gubitka.

Izbor ove funkcije najčešće zavisi od tipa problema koji se rješava, jer priroda ulaza i izlaza mreže može da diktira najpogodniji izraz funkcije gubitka (na primjer, ukoliko je poznato da je semantika izlaza slika, tada se koristi drugačija funkcija gubitka u odnosu na slučaj da je semantika izlaza bila jedinični vektor neke klase pri procesu klasifikacije). Dakle, specifičnost problema koji se rješava, diktira izbor funkcije gubitka i, u određenoj mjeri, izbor pristupa obučavanju[5].

U poglavlju 2.1.3 objašnjen je proces propagacije unazad, pri čemu je za funkciju greške odabrana **funkcija srednjekvadratne greške** (eng. *Mean Square Error*, MSE), opisana jednačinom (2.9). Ova funkcija se takođe naziva L2 gubitak. Ova funkcija se vrlo često koristi i primjenjiva je na veliki broj problema, dominantno na klasifikaciju.

[5] Načini obučavanja biće detaljno obrađeni u poglavlju 3.2.

Pored ove funkcije, često se, u klasifikacionim primjenama kada su vrijednosti ulaza i izlaza numerički manje vrijednosti, primjenjuje i **funkcija srednje apsolutne razlike** (eng. *Mean Absolute Error*, MAE), definisana jednačinom (2.39).

$$E(X,\theta) = \frac{1}{N}\sum_{i=0}^{N-1}|\hat{y}_i - y_i| \tag{2.39}$$

Za klasifikacione probleme, ponekad se koristi **prelomni gubitak** (eng. *hinge loss*), najčešće u slučaju mašina sa vektorima nosačima (eng. *Support Vector Machine*, SVM) koje, tehnički, ne predstavljaju neuronske mreže. Ova funkcija gubitka nalazi svoju primjenu u specifičnim slučajevima kada pokazuje bolje rezultate pri klasifikaciji [48]. Ova funkcija gubitka prikazana je jednačinom (2.40).

$$E(X,\theta) = \frac{1}{N}\sum_{i=0}^{N-1}\max(0, 1-\hat{y}_i y_i) \tag{2.40}$$

Huberova funkcija gubitka, opisana jednačinom 2.41 [45], najčešće se koristi pri regresiji, budući da je manje osjetljiva na individualne odmjerke koji leže daleko od ostalih u slučaju srednjekvadratne greške, jer tretira greške kvadratno samo u specifičnom intervalu, određenom koeficijentom δ.

$$E(X,\theta) = \frac{1}{N}\sum_{i=0}^{N-1} L_\delta(\hat{y}_i, y_i),$$
$$za\ \ L_\delta(\hat{y}_i, y_i) = \begin{cases} \frac{1}{2}(\hat{y}_i - y_i)^2, & |\hat{y}_i - y_i| \le \delta \\ \delta|\hat{y}_i - y_i| - \frac{1}{2}\delta^2, & |\hat{y}_i - y_i| > \delta \end{cases} \tag{2.41}$$

Daleko najkorištenija funkcija gubitka je **međuentropijska funkcija gubitka** (eng. *cross-entropy loss*) [49], koja predstavlja metriku količine dijeljenih informacija između dvaju vektora. Ova funkcija, data jednačinom (2.42), je logaritamska funkcija gubitka čija vrijednost raste kako izlaz mreže (predviđena vjerovatnoća klase) divergira od labele. Prednost ove funkcije gubitka je u tome što kažnjava značajna odstupanja više od malih, posebno penalizujući slučajeve kada mreža sa sigurnošću daje pogrešno predviđanje.

$$E(X,\theta) = \frac{1}{N}\sum_{i=0}^{N-1} -(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)) \tag{2.42}$$

Jednačina (2.42) daje slučaj binarne klasifikacije, ali u slučaju višeklasne klasifikacije, jednačina (2.43) je pogodnija, pri čemu je $\hat{y}_{i,c}$ predviđena opservacija i klase c, a $y_{i,c}$ očekivani binarni indikator vjerovatnoće (višeklasne labele tipično označavaju sa 0 ili 1 pripadnost svakoj od klasa, pa je na svakom elementu labele binarni indikator pripadnosti klasi) da je observacija i klase c.

$$E(X,\theta) = \frac{1}{N}\sum_{i=0}^{N-1} -\sum_{c=0}^{N-1} y_{i,c}\log\left(\hat{y}_{i,c}\right) \tag{2.43}$$

Međuentropijska funkcija gubitka, iako naizgled upotrebljiva samo za klasifikaciju, pored klasifikacije nalazi svoju primjenu i pri obučavanju konvolucionih autoenkodera, generativnih suparničkih mreža i različitih arhitektura za generisanje slike i audio sadržaja.

Pored ovih funkcija gubitka, koriste se i mnoge druge, od kojih su najčešće: Kullback-Leibler divergencija (relativna entropija), Poasonova raspodjela, kosinusna blizina (skalarni proizvod), multinomialni logistički gubitak, kontrastivni gubitak i gubitak sa trojkama (eng. *triplet loss*) [50] [51] [52]. Naravno, kako je već rečeno, moguće je definisati bilo kakvu funkciju gubitka, pri čemu je neophodno funkciju konstruisati na takav način da njena optimizacija utiče na podešavanje parametara mreže u željenom smjeru.

2.5 OPTIMIZATORI

Pri korištenju propagacije unazad, najčešće kod neke varijante gradijentskog spuštanja (eng. *gradient descent*), pojavljuju se optimizacioni problemi koji uveliko utiču na konvergenciju mreže i njenu tačnost, a nemaju veze sa skupom za obučavanje i arhitekturom mreže, već sa pristupom ažuriranju težina. Odabir optimizacionog algoritma, odnosno algoritma za ažuriranje težina, značajno utiče na performanse mreže i može biti presudan ne samo po pitanju trajanja obučavanja već i po pitanju konačne tačnosti mreže.

Kako cijela mreža može da se opiše jedinstvenom jednačinom sa velikim brojem slobodnih parametara koji određuju njen konačan oblik, izbor ovih parametara praktično određuje oblik funkcije mreže koja preslikava ulazni prostor u izlazni. S druge strane,

optimizator traži optimalnu kombinaciju parametara mreže kako bi funkcija mreže davala idealnu vrijednost, s obzirom na odabrane funkcije gubitka. Drugim riječima, optimizator traži minimum u prostoru parametara funkcije mreže, pri čemu je oblik tog prostora određen funkcijom gubitka i arhitekturom mreže. Prema tome, jednačine i ilustracije koje slijede tiču se formulacije jednačina optimizatora s obzirom na proces minimizacije vrijednosti funkcije greške.

Čest problem koji se javlja pri procesu optimizacije je problem odabira koeficijenta učenja, gdje niske vrijednosti mogu značajno usporiti proces obučavanja, dok visoke vrijednosti mogu dovesti do oscilacije oko minimuma ili čak do divergencije [53]. Ovo se često rješava definisanjem funkcije za smanjenje koeficijenta učenja sa brojem proteklih epoha[6]. Obično ne postoji zavisnost između skupa za obučavanje i ove promjene koeficijenta učenja, pa je problem potrebno riješiti na nivou optimizatora [40] [54]. Pored ovoga, isti koeficijent učenja se primjenjuje pri ažuriranju svih parametara, što ne mora biti poželjno svojstvo, budući da se neka obilježja potencijalno pojavljuju rjeđe i ažuriranja parametara koji njih opisuju treba da budu značajnija (ili drugačije skalirana). U slučaju da je dimenzionalnost optimizacionog prostora vrlo visoka, pojavljuje se problem konvergencije ka nekom od lokalnih minimuma funkcije, posebno u slučaju vrlo nekonveksnih funkcija. Štaviše, pored minimuma, čest uzrok lokalne konvergencije su sedlaste plohe koje se pojavljuju samo kod višedimenzionalnih funkcija [55].

Ovdje su navedeni najčešće korišteni optimizatori i njihova upotreba i prednosti. Njutnova metoda i slični pristupi, koji postavljaju nepraktične zahtjeve za izračunavanje, nisu analizirani. Pored toga, navedeni optimizatori predstavljaju modifikacije stohastičkog gradijentskog spuštanja (eng. *Stochastic Gradient Descent*), koji ažuriranja rade ne nad cijelim skupom već u iteracijama nad individualnim parovima iz skupa za obučavanje. O ovom pristupu će biti više riječi u poglavlju 3.1.

Često se pri optimizaciji javlja problem da (stohastičko) gradijentsko spuštanje nije u stanju da konvergira na upotrebljiv način (idealno, ka globalnom minimumu) kada se površina više zakrivljuje po jednoj dimenziji nego po ostalima. U ovim slučajevima, algoritam osciluje između suprotnih kosina sa vrlo sporim napretkom ka minimumu. **Impuls** (eng. *momentum*) [56] je pristup koji ubrzava konvergenciju u traženom smjeru i prigušuje oscilacije dodajući na

[6] Jedna epoha predstavlja prolazak kroz sve elemente iz skupa za obučavanje. Ovo je detaljno razrađeno u poglavlju 3.2.

trenutni delta vektor skaliranu vrijednost delta vektora iz prethodne iteracije. Jednačina (2.44) opisuje ažuriranje parametara bez impulsa, a jednačine (2.45) i (2.46) opisuju ažuriranje parametara, na jednom od parametara θ, gdje je $E(\theta)$ ekvivalentno sa $E(X, \theta)$, radi jednostavnosti zapisa.

Impulsni (momentum) član, γ, se obično postavlja na 0.9 ili neku blisku vrijednost.

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta E(\theta) \tag{2.44}$$

$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta E(\theta) \tag{2.45}$$

$$\theta_{t+1} = \theta_t - v_t \tag{2.46}$$

U ovom slučaju, optimizator se ubrzava kako se kreće niz strminu, ali budući da impulsni član ima efekat nagomilavajućeg impulsa, ovaj pristup optimizaciji povremeno rezultuje izlaskom iz minimuma pod uticajem impulsa. **Nesterovljev ubrzani gradijent** (eng. *Nesterov Accelerated Gradient*, NAG) [57] [58] rješava ovaj problem predviđajući buduću vrijednost parametra na osnovu impulsa, te kompenzirajući za tu promjenu. Prema tome, vrijednost se ažurira ne na osnovu trenutnih parametara, već na osnovu aproksimirane buduće vrijednosti parametara. Ažuriranje parametara na osnovu Nesterovljevog ubrzanog gradijenta dato je jednačinom (2.47).

$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta E(\theta - \gamma v_{t-1}) \tag{2.47}$$

Umjesto računanja trenutnog gradijenta i skakanja u smjeru ažuriranog akumuliranog gradijenta, što je slučaj kod impulsnog pristupa, NAG prvo skače u smjeru prethodnog akumuliranog gradijenta, mjeri gradijent i pravi korekciju što zajedno čini cijeli korak NAG ažuriranja. Ovo anticipirajuće ažuriranje onemogućava prevelike skokove i pokazuje odlične rezultate kod rekurentnih neuronskih mreža na velikom broju zadataka [59].

Prirodna nadogradnja na NAG, s obzirom na formulisane probleme u optimizaciji, je da se, s obzirom na važnost parametra, vrijednosti ažuriraju u većoj ili manjoj mjeri. **Adagrad** [60] je algoritam koji pravi manja ažuriranja za parametre vezane za obilježja koja se češće pojavljuju, a veća ažuriranja (veće koeficijente učenja) za parametre vezane za rjeđa obilježja. Iz ovog razloga, Adagrad je pogodan za prorijeđene podatke [61] [62]. Jednačina (2.48) opisuje ažuriranje parametara Adagrad algoritmom, gdje je $G_t \in \mathbb{R}^{d \times d}$ dijagonalna matrica kod koje su dijagonalni elementi sa indeksima i, i, sume kvadrata gradijenta u odnosu na θ_i gdje i uzima

vrijednosti do koraka u vremenu t, a ε je član za izglačavanje (radi izbjegavanja dijeljenja s nulom).

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t + \varepsilon}} \nabla_\theta E(\theta_t) \tag{2.48}$$

Glavna prednost Adagrad pristupa je u tome što se eliminiše potreba za manuelnim podešavanjem koeficijenta učenja α. Većina implementacija sa Adagrad optimizacijom ostavljaju koeficijent učenja na 0.01, iako algoritam dozvoljava izmjene. S druge strane, mana ovog algoritma je akumulacija kvadrata gradijenata u djeliocu, što dovodi, uz dovoljno iteracija, do smanjenja koeficijenta učenja na infinitezimalno male vrijednosti, nepraktične za dalje obučavanje.

Adadelta [63] je algoritam koji pokušava da riješi pomenuti problem ograničavajući prozor akumuliranih gradijenata na fiksnu veličinu. Umjesto čuvanja n prethodnih kvadrata gradijenata, suma gradijenata se definiše kao cureći (eng. *decaying*) prosjek svih prethodnih gradijenata. Tako trenutni prosječni gradijent u trenutku t zavisi samo od prethodne prosječne vrijednosti i trenutne vrijednosti gradijenta, prikazano jednačinom (2.49), gdje je sa $E[g^2]_t$ označen cureći prosjek u trenutku t.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)(\nabla_\theta E(\theta_t))^2 \tag{2.49}$$

Na osnovu ovoga, jednačina (2.48) može da se preformuliše na Adadelta jednačinu (2.50).

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \varepsilon}} \nabla_\theta E(\theta_t) \tag{2.50}$$

Striktno posmatrano, napisana jednačina više odgovara **RMSprop** algoritmu (nepublikovan algoritam, često korišten u literaturi), dok stvarna Adadelta zahtijeva podešavanje mjernih jedinica [63]. Suštinski, oba pristupa su vrlo slična i ne daju značajno drugačije rezultate.

Slično kao prethodna dva pristupa, **adaptivna estimacija impulsa** (eng. *Adaptive Moment Estimation*, Adam) [64] je algoritam koji izračunava adaptivne koeficijente učenja za svaki parametar, ali uz čuvanje eksponencijalno curećeg prosjeka kvadrata prethodnih vrijednosti gradijenata, on čuva i eksponencijalno cureći prosjek prethodnih gradijenata, slično

kao algoritam sa impulsnim članom. Adam preferira ravne minimume [65]. Jednačine (2.51)-(2.54) opisuju Adam optimizator.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \qquad g_t = \nabla_\theta J(\theta_t) \tag{2.51}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2.52}$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.53}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon} \widehat{m}_t \tag{2.54}$$

Faktori β_1 i β_2 obično uzimaju vrijednosti bliske 0.9 i 0.999, dok je ε obično izrazito mala vrijednost reda veličine 10^{-8}. Varijacija na ovaj pristup je i AdaMax koji zamjenjuje l_2 normu iz Adam algoritma l_∞ normom, jer opštija l_p norma postaje numerički nestabilna za veće vrijednosti p. Pokazuje se da izraz sa l_∞ normom konvergira na jednačinu (2.55) [64].

$$v_t = \beta_2^\infty v_{t-1} + (1-\beta_2^\infty) g_t^\infty = \max(\beta_2 v_{t-1}, |g_t|) \tag{2.55}$$

Jasno je da je Adam kombinacija impulsnog člana i RMSprop algoritma, ali pokazuje se i da je Nesterovljev ubrzani gradijent superioran u odnosu na sam optimizator sa impulsnim članom. Zbog toga je konstruisana **Nesterov-ubrzana adaptivna estimacija impulsa** (eng. *Nesterov-accelerated Adaptive Moment Estimation*, Nadam) [66] koja kombinuje Adam i NAG modifikujući impulsni član sa konačnim oblikom pravila za ažuriranje datim jednačinom (2.56).

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon} \left(\beta_1 \widehat{m}_t + \frac{(1-\beta_1) g_t}{1-\beta_1^t} \right) \tag{2.56}$$

Tokom upotrebe ovih optimizatora pojavili su se problemi sa konvergencijom ka optimalnom rješenju na različitim problemima (između ostalog kod prepoznavanja objekata [67] i automatskog prevođenja [68]), gdje se pokazalo da je gradijentsko spuštanje sa impulsnim članom efikasnije. Ovaj problem je prepoznat i formalizovan i suštinski se svodi na kratkoročno ograničenje memorisanja gradijenata zbog eksponencijalnog curenja. Predloženo rješenje je **AMSGrad** algoritam koji koristi maksimum kvadriranih gradijenata iz prošlosti umjesto eksponencijalno cureće prosječne vrijednosti [69]. Modifikacija u odnosu na Adam algoritam data je jednačinom (2.57).

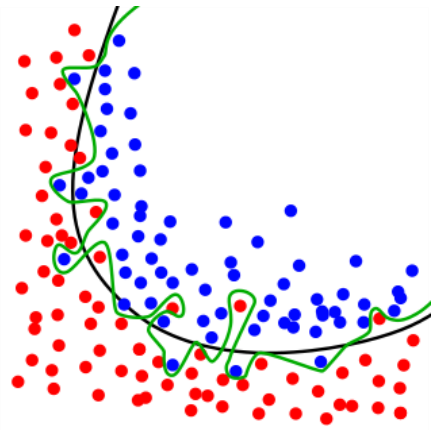
$$\hat{v}_t = \max(\hat{v}_{t-1}, v_t) \tag{2.57}$$

U većini praktičnih aplikacija Adam i Nadam se pokazuju kao izuzetno praktični i daleko efikasniji od samog gradijentskog spuštanja. Obično se ovi optimizatori biraju prvo, a u slučaju neuspješne konvergencije izvode eksperimenti sa ostalima. U većini slučajeva, Adam, Nadam i RMSprop daju slične rezultate, konzistentno bolje od gradijentskog spuštanja, a u većini slučajeva bolje od gradijentskog spuštanja sa impulsom.

2.6 PREOBUČAVANJE, GENERALIZACIJA I REGULARIZACIJA

Pri obučavanju neuronskih mreža obično je cilj da se mreža obuči na takav način da ulazni skup podataka generalizuje i na podatke koji joj nisu prezentovani pri obučavanju. Generalizacija podrazumijeva da mreža koduje znanje o ulaznom skupu sa minimalnim brojem parametara, bez adaptacije na specifičnosti skupa za obučavanje. Ukoliko mreža postane veoma dobra kada operiše nad skupom za obučavanje, a pokazuje loše performanse pri validaciji ili testiranju, kaže se da je mreža overfitovala ili da je preobučena (pretrenirana).

Mreža koja dobro generalizuje ulazni skup potencijalno pravi greške na tom skupu, ali je u stanju da pouzdano i sa visokom tačnošću obrađuje podatke koji se nisu našli u skupu za obučavanje i pokazuje dobre performanse, bez obzira na to da li joj je ulazni vektor dat ranije (za vrijeme obučavanja) ili ne. Slika 2.9 ilustruje razliku između mreže koja dobro generalizuje i mreže koja je preobučena.



Slika 2.9 [70] *– Generalizacija i overfiting. Fitovana kriva koja razdvaja dva skupa na takav način da je njena jednačina jednostavnija, a da pri tom dozvoljava greške bolje generalizuje skup od fitovane krive sa vrlo kompleksnom jednačinom koja ne dozvoljava greške. Preobučena mreža fituje samo skup za obučavanje i tipično ne daje dobre rezultate za ulaze koji joj ranije nisu dati.*

U praksi, pri obučavanju mreža sa manjim brojem parametara lakše je izbjeći problem overfitovanja (eng. *overfitting*), jer manji broj parametara onemogućava mreži da nauči kompleksniju raspodjelu. S druge strane, mreže sa velikim brojem parametara mogu da nauče cijeli skup za obučavanje, bez ikakvog „razumijevanja“ (suštinski, bez generalizacije) sadržaja koji je doveden na ulaz.

Očigledno, rješenje nije u korištenju manjih mreža, jer se kompleksnija semantika može kodovati u dodatnim parametrima. Štaviše, u slučaju da mreža ne posjeduje dovoljan broj parametara, ona potencijalno neće moći da opiše pojavu koju treba da modeluje i pojaviće se suprotan problem: nedovoljno obučavanje ili anderfitovanje (eng. *underfitting*). Ovo je garantovano slučaj kada kompleksnost problema premašuje kapacitet mreže da ga modeluje.

Glavni problem pri konstrukciji i obučavanju neuronskih mreža za praktične svrhe je u omogućavanju generalizacije. Pristupi rješavanju ovog problema biće obrađeni kasnije, ali, kako bi se upotpunio matematički model neuronskih mreža, neophodno je prije toga uvesti koncept regularizacije.

Regularizacija je statistički mehanizam koji izglačava fitovanje, odnosno uklanja ili umanjuje pojedinačna odstupanja, na nivou skupa za obučavanje. Regularizacija je širok pojam koji podrazumijeva veliki broj tehnika koje primoravaju mrežu da preferira jednostavnije fitove u odnosu na kompleksnije (jednostavniji fit podrazumijeva jednostavniju fitovanu krivu). Iako postoji veliki broj funkcija koje se mogu koristiti za regularizaciju, ovdje su predstavljene samo dvije, bazirane na l_1 i l_2 normiranju, konkretno L1 i L2 regularizacije.

Neka je generalizovana funkcija gubitka data jednačinom (2.58). Tada se, za individualne parametre $w \in \theta$, L1 regularizacija nad tom funkcijom može dati jednačinom (2.59), a L2 regularizacija jednačinom (2.60), gdje je član X izbačen jer rezultat funkcije ne zavisi od njega.

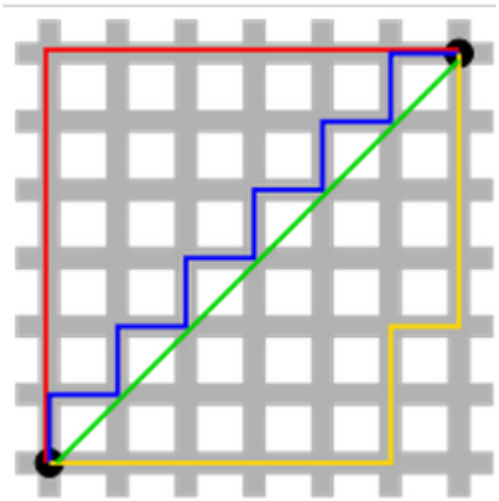
$$E(\theta) = \frac{1}{N}\sum_{i=0}^{N-1} L(\hat{y}_i, y_i) \tag{2.58}$$

$$E(\theta) = \frac{1}{N}\sum_{i=0}^{N-1} L(\hat{y}_i, y_i) + \frac{\lambda}{N}\sum_{w\in\theta} |w| \tag{2.59}$$

$$E(\theta) = \frac{1}{N}\sum_{i=0}^{N-1} L(\hat{y}_i, y_i) + \frac{\lambda}{2N}\sum_{w\in\theta} w^2 \tag{2.60}$$

Ovdje je λ koeficijent regularizacije kojim se ograničava uticaj regularizacionog člana na gradijente. Jednačine za ažuriranje parametara θ ostaju iste, pri čemu regularizacioni član utiče na izračunavanje gradijenata [71] [72]. Suštinski, povećavanjem regularizacionog koeficijenta, smanjuje se odstupanje promjena težina od prosjeka pri ažuriranju i na taj način promjene težina se čuvaju bliskim nuli. Naravno, ukoliko je ova vrijednost prevelika, mreža može biti spriječena da uči bilo kakva obilježja. Ovaj koeficijent se bira tako da se mreža spriječi u učenju sitnih detalja specifičnih za pojedinačne uzorke na ulazu, a da pri tom zadrži mogućnost učenja generalnijih obilježja. Koeficijent regularizacije, slično kao i koeficijent učenja i bilo koji drugi hiperparametar, može da se mijenja za vrijeme obučavanja, ali, tipično, koeficijent regularizacije ostaje isti u toku obučavanja.

Postoji nekoliko glavnih razlika između ove dvije vrste regularizacije, od kojh je vrlo često najznačajnija jedinstvenost rješenja [73]. Naime, L1 regularizacija ne proizvodi nužno jedinstveno rješenje, dok L2 garantuje jedinstvenost rješenja. Ovo važi za prostor proizvoljne dimenzionalnosti (veće od 1). Slika 2.10 ilustruje ovu razliku u dvodimenzionom diskretnom prostoru.



Slika 2.10 – L1 (isprekidano) i L2 (dijagonalno) pri nalaženju najkraćeg rastojanja između dvije tačke (Manhatan distanca i Euklidska distanca, respektivno) u dvodimenzionom diskrentom prostoru. Vidi se da samo L2 daje jednoznačno rješenje.

Za razliku od L2 regularizacije, L1 ima osobinu da produkuje rijetke koeficijente (odnosno, veliki broj koeficijenata je nula ili blizak nuli, dok su ostali koeficijenti relativno velike vrijednosti). Ovo je poželjna osobina kada je potrebno naglasiti značajne koeficijente, i to je slučaj kada se L1 preferira u odnosu na L2.

S druge strane, $l_1$norma nema analitičko rješenje, dok $l_2$norma ima, pa se $l_2$norma može računati efikasno i samim tim daje bolje performanse pri izračunavanju.

U principu, za regularizaciju može da se koristi bilo koja norma, uključujući i maksimum, p-normu, normu traga (eng. *trace norm*) i Hemingovu distancu, ali ove norme pri regularizaciji su izuzetno rijetke i obično se bira L1 ili L2 regularizacija, sa generalnom preferencom ka L2, osim u specifičnim slučajevima, kada je poznato da se pri optimizaciji pojavljuju važni rijetki koeficijenti.

2.7 TIPOVI MREŽA

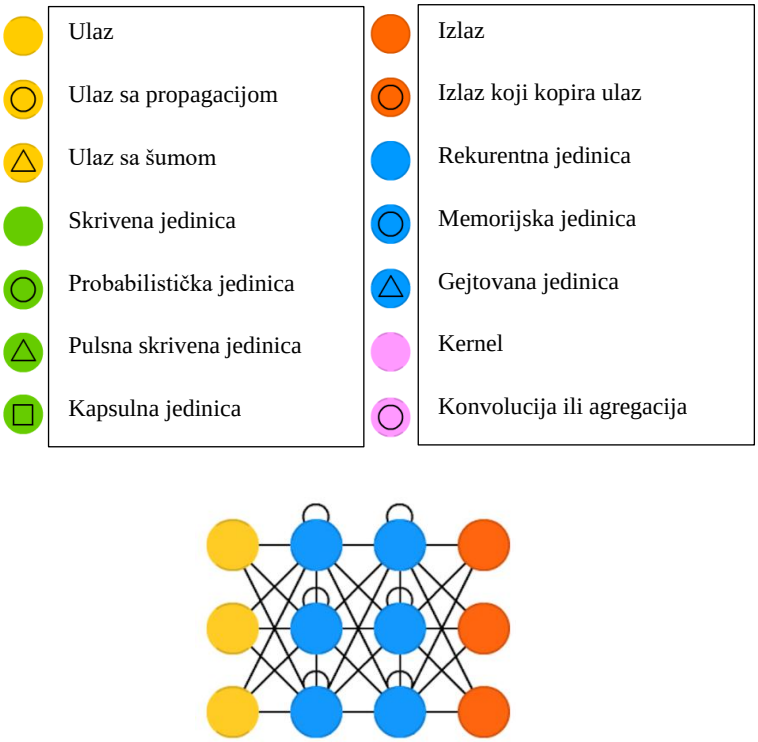
Tokom razvoja oblasti mašinskog učenja pojavile su se određene klase arhitektura vještačkih neuronskih mreža koje su se pokazale kao vrlo efikasne za rješavanje određenih specifičnih vrsta problema. Naravno, svaka od ovih klasa može posebno da se adaptira za specifičnu upotrebu, ali individualne adaptacije se i dalje mogu svrstati u neku od osnovnih klasa neuronskih mreža. Ovdje je data klasifikacija najčešće korištenih arhitektura neuronskih mreža sa nekoliko generalizovanih primjena [74], ponovo kao osnova za metodologiju razvoja predložene mreže za klasifikaciju.

Mreže sa propagacijom unaprijed (eng. *Feed Forward Neural Network*, FFNN) i klasični perceptroni predstavljaju jednostavne slojevite arhitekture opisane u prethodnom poglavlju. Ove mreže prihvataju vektor na ulaznim neuronima, te propagiraju vrijednosti kroz skrivene slojeve ka izlaznim slojevima[7] metodom propagacije unaprijed opisane ranije. Teoretski, ove mreže mogu da modeluju bilo koju zavisnost između ulaza i izlaza, što se u praksi često, s obzirom na postojeće metode obučavanja, pokazuje kao neizvodljiv zadatak. Bez obzira na to, ove mreže mogu da se koriste u relativno jednostavnim slučajevima klasifikacije ili detekcije sa zadovoljavajućom tačnošću [75].

Rekurentne neuronske mreže (eng. *Recurrent Neural Network*, RNN) predstavljaju slojevite mreže sa jedinicama koje prave rekurentne konekcije (odnosno formiraju petlje). Kod ovih mreža, model vještačkog neurona je takav da individualni neuroni čuvaju informacije o istoriji svojih izlaza i pri aktivaciji na ulaz dodaju i svoj prethodni izlaz (odnosno izlaz generisan pri prethodnoj inferenciji). Ovo omogućava rekurentnim neuronskim mrežama čuvanje temporalnih informacija, odnosno povezanost sa prethodnim rezultatima izvršavanja. Na ovaj način, rekurentne mreže mogu da se koriste za predikciju vremenskih serija i rješavanje problema čiji konstituentni podaci imaju vremensku zavisnost. Ovo nije jedini način da se vremenska zavisnost modeluje, ali je računski jeftin, jer isti neuron čuva podatak o svojoj prošlosti, umjesto da se ti podaci čuvaju u nekoj odvojenoj strukturi. S druge strane, činjenica da individualni neuroni čuvaju ove podatke dovodi do takozvanog problema nestajućeg gradijenta, kao i do problema eksplodirajućeg gradijenta[8], zavisno od izbora aktivacione funkcije. Sa lošim ili pogrešnim izborom aktivacione funkcije, podaci iz prošlosti mogu da postanu irelevantni za izračunavanje zbog smanjenja doprinosa rekurentne veze. Vrijedi

[7] U Glavi 3 objašnjene su arhitekture neuronskih mreža koje sadrže više izlaznih slojeva. Ovo je relativno česta pojava pri projektovanju neuronskih mreža, obično sa svrhom poboljšavanja procesa obučavanja.
[8] Česti problemi pri obučavanju neuronskih mreža, uključujući eksploziju gradijenata, objašnjeni su u Glavi 3.

napomenuti da se ove mreže ne moraju nužno koristiti nad vremenski zavisnim skupovima podataka, već na bilo kom skupu podataka koji se može raspodijeliti u sekvencu međusobno zavisnih susjednih podataka. Budući da ne postoji ograničenje za veličinu ulaza na ovaj način, ovakve mreže, i njihove podklase, često se koriste za prepoznavanje glasa, tumačenje teksta i detekciju obrazaca u serijama podataka [76]. Uprošćena arhitektura rekurentnih neuronskih mreža data je Slikom 2.12.



Slika 2.12 [74] *– Definicije simbola (gore), arhitektura rekurentne neuronske mreže (dole)*

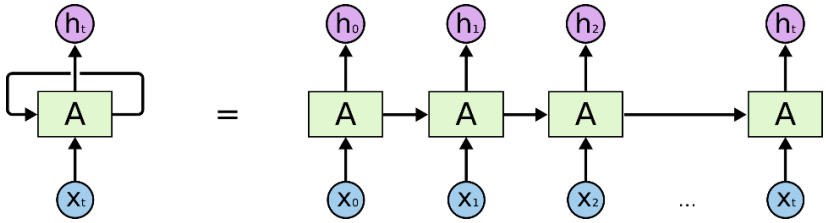
Mreže sa dugotrajnom kratkoročnom memorijom (eng. *Long Short-Term Memory*, LSTM) [77] su adaptirane rekurentne neuronske mreže koje dijelom rješavaju probleme nestajućeg i eksplodirajućeg gradijenta modifikacijom osnovnog modela vještačkog neurona. Osnovni model neurona kod LSTM mreža modifikovan je uvođenjem gejta za zaboravljanjem i posebne memorijske ćelije na nivou vještačkog neurona. Suštinski, većina uspješnih implementacija rekurentnih neuronskih mreža koristi neku vrstu LSTM mreže, jer ove mreže vrlo dobro rješavaju pomenuti problem nestajućih i eksplodirajućih gradijenata.

2.7.1 Mreže sa dugotrajnom kratkoročnom memorijom

Iako mreže sa dugotrajnom kratkoročnom memorijom nisu direktan predmet ove disertacije, ovdje su date jednačine za jednu implementaciju LSTM ćelije, u svrhu prikazivanja mogućnosti modelovanja neuronske mreže na nivou neuronskih jedinica. Za svrhu klasifikacije

slika, kakva je klasifikacija aero-snimaka, ovakav pristup je rijetko neophodan, no može biti od koristi ukoliko se klasifikacija vrši nad individualnim frejmovima video zapisa, gdje postoji vremenska korelacija između potencijalnih piksela ulaza. Takođe, ovdje je dat pristup razmotavanju LSTM mreža sa ciljem uklanjanja vremenski rekurentnih veza.

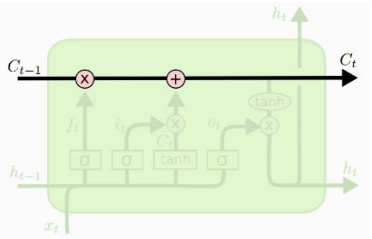
Rekurentne neuronske mreže modifikuju model neurona dodajući na njega jednu cikličnu vezu, odnosno vežući izlaz neurona na njegov ulaz, kao dodatak na prethodno izložen matematički model. Ovako modifikovan vještački neuron može se predstaviti kao na Slici 2.13.



Slika 2.13 [78] – *Model rekurentnog neurona i njegov nerekurentni ekvivalent, gdje* x_t *predstavlja ulaz u jedinicu* A *u trenutku* t, *a* h_t *njen izlaz u trenutku* t.

Vrlo značajna osobina rekurentnih neuronskih mreža je mogućnost njihovog vremenskog razmotavanja. Naime, moguće je svaki rekurentni vještački neuron pretvoriti u seriju klasičnih neurona, gdje se na ulaz svakog dovodi uzorak iz različitog trenutka u vremenu. Ovo, sa izračunske strane, unosi dodatne memorijske zahtjeve, jer se broj varijabli povećava sa brojem koraka u prošlost koji se uzimaju u obzir. Teoretski, rekurentni neuroni mogu da prave beskonačan broj koraka u prošlost, što nije smisleno sa stanovišta razmotavanja, budući da je fizička memorija ograničena, no postoji logička granica gdje dodatni koraci u prošlost prestaju imati smisla. Štaviše, LSTM mreže pokušavaju da riješe ovaj problem, dodatno modifikujući model individualnog neurona. Iako RNN mreže teoretski mogu da nauče bilo kakvu vremensku zavisnost, pokazuje se da u praksi postoje ozbiljne prepreke ka tome [79].

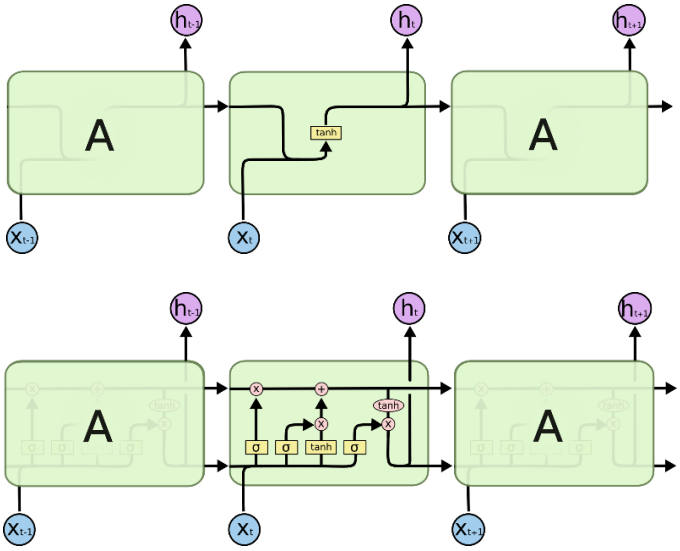
Ključna ideja za LSTM mreže je stanje memorijske ćelije koje se propagira kroz vrijeme odvojeno od podataka. Zbog ovoga, LSTM neuron ima dva izlaza: stanje ćelije i modifikovani izlaz. LSTM mreža može da dodaje ili uklanja stanje ćelije, što je regulisano gejtovima. Gejtovi se tipično sastoje od sigmoidne aktivacione funkcije i množačkog čvora. Slika 2.14 predstavlja propagaciju stanja kroz vrijeme.



Slika 2.14 – Propagacija stanja LSTM ćelije, kroz vrijeme (izlaz iste ćelije u prethodnom vremenskom trenutku prosljeđuje se na trenutni ulaz ćelije)

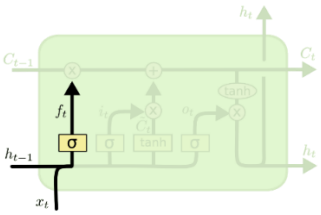
Posmatrajući razmotanu RNN mrežu sa proizvoljnom aktivacionom funkcijom, kao na Slici 2.15, može se primijetiti relativno jednostavna struktura. S druge strane, LSTM mreža sadrži daleko kompleksniji model rekurentnog neurona. Vrijedi napomenuti da na Slici 2.15 svaka linija predstavlja ulazni vektor, jer se simbolom *A* označava ne jedan neuron, već svi neuroni u datom rekurentnom sloju (ili, drugačije gledano, *A* označava neuron koji na ulazu prihvata vektor i svoje operacije izvodi nad individualnim elementima).

Pri formiranju modela LSTM neurona, prvi korak je odbacivanje dijela informacija iz stanja ćelije korištenjem gejt jedinice. Gejt jedinica posmatra zbir težinskom matricom modifikovanog vektora ulaza neurona u trenutku t i težinskom matricom modifikovani izlaz istog neurona sa prethodnog trenutka u vremenu ($\boldsymbol{x}_t$ i $\boldsymbol{h}_{t-1}$, respektivno) i primjenjuje sigmoidnu aktivacionu funkciju na njih, čiji izlaz je vrijednost u intervalu $[0, 1]$. Ovaj izlaz se množi sa prethodnim stanjem memorijske ćelije $\boldsymbol{C}_{t-1}$, efektivno brišući ili umanjujući individualne vrijednosti u vektoru memorijske ćelije $\boldsymbol{C}_t$ na osnovu izlaza sigmoidnog gejta. Ovo je, sa prednaponskim članom, predstavljeno Slikom 2.16 i opisano jednačinom (2.61). Ovaj, prvi gejt naziva se gejt za zaboravljanje (eng. *forget gate*).



Slika 2.15 – Razmotan model rekurentnog neurona (gore) i razmotan model LSTM neurona (dole)

Sljedeći korak je odlučivanje o tome koji dio informacija će biti sačuvan u memorijskoj ćeliji. Ovaj se korak sastoji od dva dijela: primjena ulaznog gejta (eng. *input gate*) koji odabira vrijednosti koje će biti ažurirane i konstrukcija vektora kandidatskih vrijednosti $\widetilde{\boldsymbol{C}}_t$, za upis u memorijsku ćeliju.



Slika 2.16 – Gejt za zaboravljanje

Kako bi se dobila nova vrijednost $\boldsymbol{C_t}$, potrebno je, nakon primjene gejta za zaboravljanje (odnosno množenja vrijednosti $\boldsymbol{C_{t-1}}$ sa međuvrijednošću $\boldsymbol{f_t}$), na vrijednost dobijenu nakon primjene gejta za zaboravljanje dodati $\boldsymbol{i_t} * \boldsymbol{C_t}$ (gdje je $*$ Adamardov proizvod, odnosno množenje elemenata član po član). Izračunavanje ovih vrijednosti dato je jednačinama (2.62) i (2.63), dok je ažuriranje memorijske ćelije dato jednačinom (2.64). Matrice $\boldsymbol{W}$ i $\boldsymbol{U}$ predstavljaju matrice težina, a vektori $\boldsymbol{b}$ prednaponske vektore [80] [81].

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) \tag{2.61}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{2.62}$$

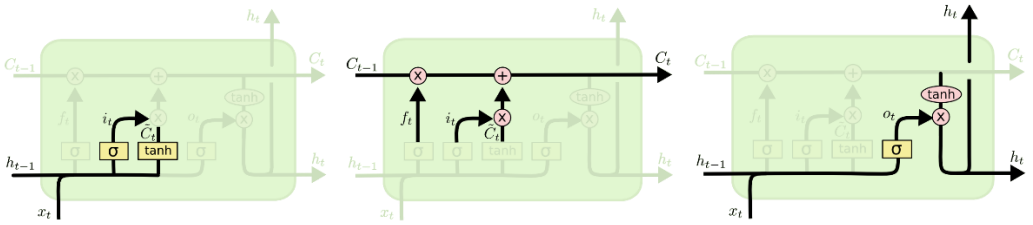
$$\widetilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{W}_C \boldsymbol{x}_t + \boldsymbol{U}_C \boldsymbol{h}_{t-1} + \boldsymbol{b}_C) \tag{2.63}$$

$$\boldsymbol{C}_t = \boldsymbol{f}_t * \boldsymbol{C}_{t-1} + \boldsymbol{i}_t * \widetilde{\boldsymbol{C}}_t \tag{2.64}$$

Konačno, potrebno je odrediti izlaz vještačkog neurona, što se dobija izračunavanjem Adamardovog proizvoda nad vrijednošću vektora sačuvanog u memorijskoj ćeliji, delinearizovanog aktivacionom funkcijom hiperbolnog tangensa, i vrijednošću ulaznog vektora delinearzovanog sigmoidnom aktivacionom funkcijom. Ovo izračunavanje dato je jednačinama (2.65) i (2.66). Slika 2.17 predstavlja individualne korake opisane datim jednačinama.

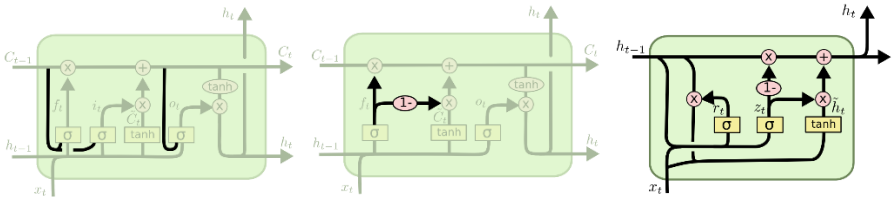
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{2.65}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(\boldsymbol{C}_t) \tag{2.66}$$



Slika 2.17 – Ulazni gejt, podešavanje memorijske ćelije, izračunavanje izlaza (redom, slijeva udesno)

Bitno je naglasiti da postoji veliki broj varijacija na osnovnu arhitekturu LSTM mreža, te da je moguće individualne neurone modelovati i na drugačiji način. Neke od varijacija [82] [83] [84] [85] [86] date su na Slici 2.18.



Slika 2.18 – Varijacije modela LSTM neurona (tamnije linije označavaju razliku u odnosu na prethodno izloženi model)

Mreže sa dugotrajnom kratkoročnom memorijom su vrlo uspješno korištene u velikom broju oblasti, od kojih su najistaknutije prepoznavanje govora, automatsko prevođenje i prepoznavanje teksta u slikama kontrolisanim klizećim prozorom. Ovo čini rekurentne neuronske mreže posebnom klasom neuronskih mreža koja se izrazito efikasno koristi u vrlo specifičnom skupu podoblasti.

■

Gejtovane rekurentne jedinice (eng. *Gated Recurrent Units,* GRU) [87] su relativno jednostavna varijacija na LSTM mreže: umjesto ulaznog gejta, izlaznog gejta i gejta za zaboravljanje, ove mreže imaju samo jedan gejt za ažuriranje (eng. *update gate*) koji preuzima funkciju sva tri gejta LSTM mreže, odnosno obavlja i brisanje i upis kao jednu operaciju. Uglavnom, ove mreže funkcionišu slično kao LSTM mreže, pri čemu daju malo bolje performanse zbog smanjenog broja potrebnih matematičkih operacija.

Neuronska Tjuringova mašina (eng. *Neural Turing Machines,* NTM) [88] je relativno rijetka implementacija neuronske mreže koja predstavlja varijaciju na LSTM mrežu pri čemu se memorijske ćelije izdvajaju iz neuronskih jedinica s ciljem postojanja memorije koju mogu da adresiraju individualni neuroni. Činjenica da je memorija adresabilna čini ovakvu mrežu Tjuring-kompletnom, odakle i dolazi njeno ime. Iako ova vrsta mreže nije česta, varijacije na ovu arhitekturu koriste se u implementacijama koje se tiču učenja s podsticajem (eng. *reinforcement learning*).

Još jedan tip rekurentne mreže je **mreža sa stanjem eha** (eng. *Echo State Network,* ESN) [89]. Ove mreže sadrže nasumično povezane neurone koji nisu grupisani u slojeve. Umjesto klasičnog očitavanja izlaza nakon postavljanja ulaza, nakon postavljanja ulaza aktivno se preuzimaju vrijednosti sa izlaza, kako mreža evoluira kroz vrijeme.

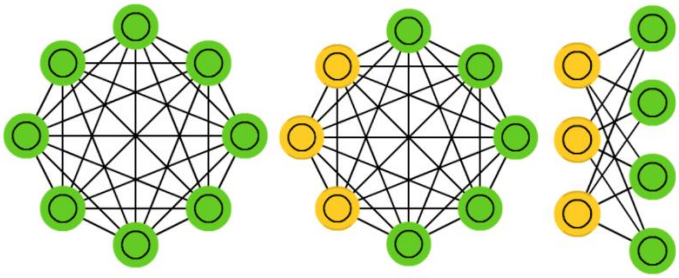
Hopfildova mreža [90] je mreža neurona koji su, umjesto grupisanja u slojeve, kako je slučaj kod FFNN mreža, svi međusobno povezani. Isti neuroni se koriste kao ulazne, skrivene i izlazne jedinice, sa značajnim varijacijama u procesu obučavanja, gdje se koristi minimizacija „temperature“ ili „energije“ mreže[9], a promjena konteksta individualnih neurona (ulazni, skriveni ili izlazni) odvija na sličan način kako se vrši ažuriranje rekurentnih neurona kod rekurentnih neuronskih mreža. Ovaj model je usko vezan za simulirano očvršćivanje (eng. *simulated annealing*) [91] i konstruisan je s ciljem korištenja u prepoznavanju obrazaca, no mreža ne konvergira uvijek, što je čini upotrebljivom u relativno malom broju slučajeva.

Bolcmanove mašine (eng. *Boltzmann Machine,* BM) [92] su mreže slične Hopfildovim mrežama, uz varijaciju da se neki od neurona proglašavaju ulaznim, a neki skrivenim. Ulazni neuroni postaju izlazni po završetku punog ažuriranja mreže. Ove mreže uče propagacijom unazad ili kontrastivnom divergencijom [93] [94]. Slično kao Markovljevi lanci, ove mreže su stohastičke. Proces obučavanja je sličan kao kod Hopfildovih mreža: postave se vrijednosti

[9] Optimizacioni član nazvan *temperatura* se često koristi u svrhu sličnu koeficijentu učenja [90] [91] [92].

ulaznih neurona i mreža se izvršava do konvergencije, sa povremenim dodavanjem na ulaze (odnosno promjenom konteksta ulaznih neurona sa ulaznih na skrivene i nazad). Aktivacija se kontroliše globalnom vrijednošću temperature, čijim smanjivanjem se smanjuje energija individualnih neurona. Ovim smanjivanjem, mreža se postepeno dovodi u stabilno stanje i, uz ispravno podešavanje parametara, dostiže ekvilibrijum. Ove mreže se pokazuju kao upotrebljive u relativno jednostavnim zadacima klasifikacije i uklanjanja šuma, pogotovo u slučaju ulaznih vektora manje dimenzionalnosti.

Ograničene Bolcmanove mašine (eng. *Restricted Boltzmann Machine,* RBM) [95] su izuzetno slične Bolcmanovim mašinama, s tom razlikom da ne postoje veze između svih neurona u mreži, već su neuroni izdvojeni u nezavisne grupe, slično kao što je slučaj sa slojevima u FFNN mrežama. Slika 2.19 ilustruje Hopfildovu mrežu, Bolcmanovu mašinu i ograničenu Bolcmanovu mašinu.

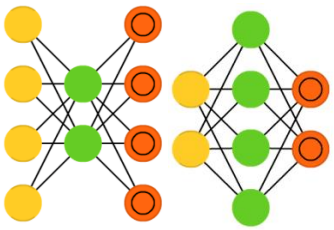


Slika 2.19 – Hopfildova mreža, Bolcmanova mašina, ograničena Bolcmanova mašina (redom, slijeva udesno)

Autoenkoderi [96], predstavljeni Slikom 2.20, su specifična podvrsta FFNN mreža, gdje je suštinska razlika između njih u načinu obučavanja. Ulaz autoenkodera i njegov očekivani izlaz u domenu labela predstavljaju iste vektore. Cilj autoenkodera je rekonstrukcija ulaznog vektora, pri čemu su skriveni slojevi uvijek manje dimenzionalnosti (odnosno sadrže manji broj neurona) u odnosu na ulazni i izlazni sloj (koji moraju biti iste dimenzionalnosti, s obzirom na pomenuti način obučavanja). Ovo omogućava autenkoderu da suštinsku reprezentaciju ulaznog vektora komprimuje čuvajući reprezentacije u vektorima manje dimenzionalnosti. Ovo znači da se ulazni prostor projektuje na prostor manje dimenzionalnosti čija je gustina informacija veća. Drugim riječima, skriveni sloj sadrži fundamentalni opis ulaznog vektora, po pretpostavci bez irelevantnih informacija.

Prorijeđeni autoenkoderi (eng. *Sparse Autoencoder,* SAE) [97], takođe ilustrovani Slikom 2.20, predstavljaju podvrstu FFNN, slično kao i autoenkoderi, pri čemu su skriveni slojevi podrazumijevano veće dimenzionalnosti. Ovo omogućava kodovanje ulaznog vektora i

njegovu projekciju na prostor veće dimenzionalnosti, gdje se, po pretpostavci, čuva više informacija o ulazu nego što je potencijalno sadržano u bilo kom individualnom ulazu. Kako bi se izbjeglo kreiranje jedinične mreže (mreže koja ulaze direktno preslikava na izlaze, odnosno mreže koja se može predstaviti jediničnom matricom), izlaz se (pri propagaciji unazad) filtrira, tako što se neke vrijednosti greške ne propagiraju unazad, gdje se odabir vrši na osnovu funkcije nazvane upravljačem prorijeđenosti (eng. *sparsity driver*).



Slika 2.20 – Autoenkoder (lijevo) i prorijeđeni autoenkoder (desno)

Dodatna vrsta autoenkodera su takozvani **varijacioni autoenkoderi** (eng. *Variational Autoencoder,* VAE) [98] koji imaju istu arhitekturu kao i klasični autoenkoderi, pri čemu se obučavaju za drugi cilj: aproksimaciju raspodjele vjerovatnoća ulaznih vektora (uzoraka). Proces njihovog obučavanja oslanja se na Bajesovu teoriju, odnosno matematiku uslovnih vjerovatnoća. Suštinski, propagacija unazad uzima u obzir međusobni (statistički) uticaj različitih čvorova i potencijalno umanjuje ili povećava gradijente s obzirom na težine koje se ažuriraju i međusobnu zavisnost čvorova uzetih u obzir.

Autoenkoderi sa uklanjanjem šuma (eng. *Denoising Autoencoders,* DAE) [99] su još jedna varijacija na osnovni autoenkoder, gdje se na ulazni vektor dodaje šum, a na izlaz dovodi originalni vektor bez dodatog šuma. Ovo omogućava autoenkoderu da nauči osobine ulaznog podatka, bez obzira na šum i tako omogući cijeloj mreži da uči opštije detalje umjesto specifičnih obilježja niskog nivoa. Ovi autoenkoderi, kako je jasno iz imena, koriste se za uklanjanje šuma i pokazali su se kao vrlo efikasni u ovoj primjeni.

Mreže sa dubokim uvjerenjem (eng. *Deep Belief Networks,* DBN) [100] su mreže koje se formiraju međusobnim povezivanjem obučenih autoenkodera različitih vrsta (najčešće RBM ili VAE), gdje svaki dodatni autoenkoder treba da nauči kako da koduje izlaz prethodne mreže. Ova tehnika poznata je kao pohlepno obučavanje (eng. *greedy training*) gdje se ne traži optimalno rješenje, već se kombinuju suboptimalna rješenja sa ciljem dobijanja zadovoljavajuće tačnosti. Ove mreže mogu se obučavati propagacijom unazad i kontrastivnom divergencijom [93] [94], te se mogu obučiti da probabilistički predstavljaju podatke, slično kao

varijacioni autoenkoderi. Ispravno obučeni modeli mogu se koristiti za generisanje novih podataka na osnovu nasumičnog ulaza, pa čak i za klasifikaciju podataka (za šta, generalno, postoje daleko efikasniji modeli).

Za specifične primjene, moguće je koristiti i **konvolucione autoenkodere** (eng. *convolutional autoencoder*) koji predstavljaju autoenkodere sa konvolucionim slojevima. Konkretnije, ovi autoenkoderi mogu da se koriste nad ulaznim podacima promjenljive veličine, jer operacija konvolucije, kako je definisano ranije, nije ograničena na jednu specifičnu veličinu ulaznog podatka, već samo na njegovu dimenzionalnost. Ovi enkoderi mogu da se upotrebljavaju i kao prorijeđeni autoenkoderi i kao autoenkoderi sa uklanjanjem šuma nad podacima čija je veličina različita od one korištene za vrijeme obučavanja.

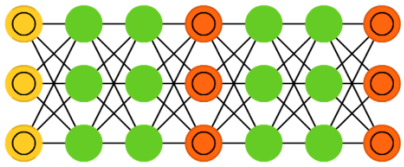
Konvolucione neuronske mreže (eng. *Convolutional Neural Network*, CNN) su daleko najkorištenija arhitektura neuronske mreže sa velikim brojem varijacija, a varijacije u arhitekturi ovih mreža su od najvećeg interesa za modernu oblast mašinskog učenja. Osnovna funkcionalnost ovih mreža opisana je ranije, a njihove arhitekturalne varijacije su detaljno razrađene u nastavku. U literaturi se često pojavljuje i termin **duboke konvolucione neuronske mreže** (eng. *Deep Convolutional Neural Networks*, DCNN) ili samo **duboke neuronske mreže** (eng. *Deep Neural Networks*, DNN) što su relativno slobodni termini za bilo koju neuronsku mrežu sa velikim brojem slojeva. Pored klasičnih konvolucionih neuronskih mreža, koje uključuju i potpuno povezane slojeve, postoje i **potpuno konvolucione mreže** (eng. *Fully Convolutional Neural Networks*, FCN) koje sadrže samo konvolucione slojeve i slojeve za agregaciju.

Dekonvolucione mreže (eng. *Deconvolutional Network*) [101], takođe poznate kao **inverzne grafičke mreže** (eng. *Inverse Graphics Network*, IGN) predstavljaju konvolucione mreže sa transponovanim konvolucijama[10], gdje uzastopni slojevi mreže postaju veći (po broju neurona) od ulaza ka izlazu. Suštinski, operacije iz konvolucionih mreža zamijenjene su svojim inverznim operacijama u dekonvolucionim mrežama. Ove mreže se najčešće koriste za generisanje podataka (obično slika na osnovu neke statističke raspodjele). Takođe, ove mreže mogu da se koriste u kombinacijama sa drugim mrežama, gdje se mogu vezivati u različite varijante konvolucionih autoenkodera, konvolucionih slaganih mreža, slaganih autoenkodera i slično.

[10] Operacija transponovane konvolucije objašnjena je u Glavi 3.

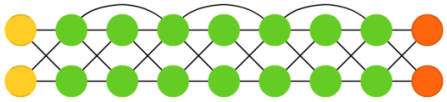
Jedna češća varijacija pomenute kombinacije arhitektura su **duboke konvolucione inverzne grafičke mreže** (eng. *Deep Convolutional Inverse Graphics Network*, DCIGN) [102], koje suštinski predstavljaju varijacioni autoenkoder sa CNN i DNN mrežama na mjestu enkodera i dekodera. Ove mreže koriste se za generisanje novih uzoraka na osnovu naučenih, kao i za generisanje kombinacija uzoraka na osnovu naučenih individualnih.

Generativne suparničke mreže (eng. *Generative Adversarial Networks*, GAN) [103] su mreže obično konstruisane sa ciljem generisanja novog sadržaja, ali sastoje se od dvaju mreža koje se obučavaju u suparničkom režimu. Ove mreže se obično sastoje od generatorske i diskriminatorske mreže, gdje generatorska mreža pokušava da generiše nove primjerke na osnovu neke ulazne raspodjele, a diskriminatorska mreža pokušava da binarno klasifikuje ulaz kao generisan ili stvaran. Ovako obučavane mreže pokazuju se kao izuzetno dobre u generisanju novog sadržaja, pogotovo kada se implementiraju u kombinaciji sa nekom varijacionom mrežom (npr. varijacionim autoenkoderom), gdje slični ulazi generišu slične izlaze. Na Slici 2.21 predstavljena je arhitektura generativne suparničke mreže, pri čemu je bitno naglasiti da ove mreže nisu značajno specifične po arhitekturi, već po načinu obučavanja.



Slika 2.21 – Generativna suparnička mreža (lijevo: generatorska mreža, desno: diskriminatorska mreža)

Duboke rezidualne mreže (eng. *Deep Residual Network*, DRN) [104] su vrlo duboke (tipično desetine ili stotine slojeva) FFNN mreže sa dodatnim vezama koje preskaču slojeve, često nazivanim preskačućim vezama (eng. *skip connection*). Ove mreže omogućavaju konstrukciju dubokih slojeva bez nastajanja problema nestajućih gradijenata, jer se na dublje slojeve direktno propagiraju i ulaz mreže i izlazi plitkih slojeva, bez gubitaka i umanjivanja. Ovakve mreže često imaju i preko stotinu slojeva. Pokazuje se da su ove mreže izrazito dobre u klasifikaciji objekata. Arhitektura ovakve mreže prikazana je na Slici 2.22.



Slika 2.22 – Duboka rezidualna mreža sa preskačućim vezama

Pored pomenutih neuronskih mreža, postoje i takozvane **pulsne neuronske mreže** (eng. *Spiking Neural Network*, SNN) [27] koje bliže oponašaju prirodne neurone i oslanjaju se na izvršavanje zasnovano na događajima i akviziciji ulaznih podataka. Ove mreže rade inferenciju i propagaciju unazad samo kada su podaci dostupni i oslanjaju se na vremenski redoslijed izvršavanja individualnih neurona, odnosno najbliže simuliraju okidanje kod bioloških neurona.

Zavisno od načina obučavanja, mogu se formirati i različite varijante pulsnih mreža, kakve su **ekstremne mašine za učenje** (eng. *Extreme Learning Machine*, ELM) [105] koje se oslanjaju na fitovanje metodom najmanjih kvadrata (što je brže od propagacije unazad, ali daje lošije rezultate) i **mašine tečnog stanja** (eng. *Liquid State Machine*, LSM) [106], koje su okidajuće mreže sa akumulirajućim memorijskim ćelijama.

Navedene arhitekture neuronskih mreža predstavljaju praktično upotrebljive tipove mreža, no za svaku od navedenih klasa neuronskih mreža postoji veliki broj varijacija u arhitekturi. Od navedenih arhitektura od interesa za ovu disertaciju su uglavnom arhitekture koje se mogu konstruisati povezivanjem slojeva različitih tipova i obučavati korištenjem neke vrste propagacije unazad, jer su to najčešće korišteni pristupi u praksi i za te tipove mreža postoji dovoljno akumuliranog znanja i iskustva da bi se mogla formirati osnovna grupa pravila za razvoj. Prema tome, od interesa su konvolucione mreže, duboke rezidualne mreže, dekonvolucione mreže, autoenkoderi i sve njihove arhitekturalne kombinacije, koje su detaljno analizirane u Glavi 4.

3 PROGRAMSKI I LOGIČKI MODEL NEURONSKE MREŽE

3.1 PREGLED

S obzirom na to da se, u praksi, neuronske mreže projektuju i konstruišu za izvršavanje na višeprocesorskim računarskim sistemima, neophodno je definisani matematički model transformisati za primjenu u stvarnom svijetu.

Ova glava daje neophodna proširenja matematičkog modela i izlaže osnovne probleme koji se mogu pojaviti pri obučavanju neuronskih mreža.

Poglavlje 3.2 opisuje različite pristupe obučavanju, kao i objašnjenje funkcionisanja gradijentskog spuštanja i njegovih praktično izvedivih varijanti. Pored toga, u ovom poglavlju predstavljeno je i opisano učenje za više zadataka, koje je suštinski mehanizam na osnovu kog je implementirano rješenje za automatsku klasifikaciju i segmentaciju zemljišta u okviru ove disertacije. Učenje za više zadataka opisano je kako bi se mogao razumjeti pristup projektovanju i obučavanju neuronske mreže iskorištene pri rješavanju ovog zadatka.

U poglavljima 3.3 i 3.4 dat je pregled osnovnih problema koji se pojavljuju pri obučavanju neuronskih mreža, njihova praktična manifestacija, kao i pristupi njihovom rješavanju. Uz preporuke za rješavanje ovih problema, dati su i načini na koje se pojava ovih problema može prepoznati u stvarnom svijetu. Ove preporuke bazirane su na postojećoj literaturi i desetinama eksperimenata izvršenim za svrhu ove disertacije. Dodatno, poglavlje 3.4 daje i primjere načina augmentacije ulaznih podataka, od kojih su neki korišteni pri realizaciji rješenja za klasifikaciju aero-snimaka.

Poglavlje 3.5 sadrži objašnjenje koncepta arhitekture neuronske mreže i enumeraciju svih slojeva neophodnih za razumijevanje predloženog rješenja i prateće metodologije razvoja. Ovdje su date i neke specifične arhitekture, na osnovu kojih je adaptiran predloženi model.

Radi potpunosti programskog modela, u poglavlju 3.6 dat je pregled različitih pristupa implementacijama na specifičnim hardverskim arhitekturama. Iako ovo nije od suštinske važnosti za implementaciju predloženog rješenja, neophodno je radi potpunosti metodologije razvoja.

Konačno, poglavlje 3.7 sadrži preporuke i pristupe za poboljšanje performansi i tačnosti klasifikacije neuronskih mreža. Ovdje su date preporuke za inicijalizaciju i podešavanje

hiperparametara mreža. Inicijalizaciona šema predstavljena ovdje iskorištena je pri obučavanju predložene mreže.

3.2 OBUČAVANJE

3.2.1 Nadgledano, nenadgledano i polunadgledano učenje

Iako postoje veoma različiti pristupi obučavanju neuronskih mreža, od kojih se mnogi uopšte ne oslanjaju na propagaciju gradijenata (na primjer, korištenje genetičkih algoritama, simulirano očvršćivanje, stohastičke metode i sl.), mašinsko učenje prepoznaje i imenuje dvije najčešće korištene klase obučavanja: nadgledano i nenadgledano učenje; oba pristupa se koriste bez obzira na to da li se podešavanje težina radi propagacijom gradijenata ili na neki drugi način.

Nadgledano učenje (eng. *supervised learning*), već opisano u poglavlju 2.1, predstavlja pristup obučavanju kod kog postoji skup podataka za obučavanje (eng. *training set*) koji se sastoji od prethodno labeliranih parova ulaz-izlaz i koji se koristi za podešavanje parametara mreže pri propagaciji unazad. Ovi skupovi podataka obično su ručno labelirani od strane stručne osobe u polju od interesa. Pri labeliranju, stručna osoba označava ili uparuje specifične ulaze u mrežu (bilo da se radi o slikama, audio zapisu, numeričkim podacima ili bilo kojoj drugoj vrsti strukturisanih podataka čija je semantika strukturisanja razumljiva labelaru, kao jedinci sa kognitivnim pristrasnostima svojstvenim ljudskim bićima) sa odgovarajućim očekivanim izlazima (labelama klasa, očekivanim slikama, zvučnim odmjercima i sl.). U idealnom slučaju, ovi se skupovi dijele na skup za obučavanje, skup za validaciju i skup za testiranje, jer je neophodno mrežu validirati nakon izmjena arhitekture (ili promjena hiperparametara), na skupu podataka koji nije korišten pri obučavanju, te je, u idealnom slučaju, potrebno, po realizaciji finalne arhitekture mreže testirati tačnost mreže na potpuno novim podacima. Tipična podjela nekog skupa ulaznih podataka je 60% za obučavanje, 20% za validaciju i 20% za testiranje. Nerijetko se skup za testiranje koristi za validaciju u omjeru: 80% za obučavanje i 20% za validaciju i testiranje. Dodatno, ukoliko je neophodan visok stepen generalizacije, tada se, pod uslovom da postoji odgovarajući broj uzoraka za obučavanje, skup dijeli i u omjeru: 10% za obučavanje i 90% za validaciju.

Pored labeliranih skupova sa parovima ulaz-izlaz, strukturisani sekvencijalni podaci, kao što su vremenske serije, odmjerci funkcija, berzanski podaci, susjedni frejmovi video zapisa i slično, mogu da se koriste pri obuci, gdje se vrši predviđanje neke vrijednosti na osnovu

prethodnih u skupu. Ovakav vid obučavanja i dalje predstavlja nadgledano učenje, jer postoji jasna definicija očekivanog izlaza za bilo koji ulaz, te se odgovarajuće upareni ulazi i izlazi mogu dobiti iz sekvencijalnog skupa podataka.

Suštinski, ovdje se razlikuju dvije vrste nadgledanog učenja: one koje rješavaju klasifikacione probleme i one koje rješavaju regresione probleme. **Klasifikacija** je proces svrstavanja ulaznih podataka u neku od definisanih kategorija (klasa), na osnovu apstraktno ili konkretno definisanog kriterijuma (ovo može biti, na primjer, klasifikacija slika objekata po nazivima ili životinja po vrsti), pri čemu se, u praksi, izlazi mreže najčešće definišu u obliku jediničnih vektora čija dimenzionalnost odgovara broju klasa i čiji indeks maksimalnog elementa odgovara indeksu klase. Naravno, ovo ne mora da bude slučaj, ali najčešće se klasifikacija radi u ovom obliku (sa izuzetkom višeklasne klasifikacije kod koje je svakom ulazu dodijeljeno više labela klasa).

S druge strane, **regresija** predstavlja proces predviđanja kontinualnih vrijednosti na osnovu ulaznog skupa, gdje elementi izlaznog vektora mreže ne moraju nužno biti binarne vrijednosti. U slučaju jednog izlaza, regresioni model projektuje neki vektor sa ulaza u skalarnu vrijednost na izlazu. Suštinski, regresija pokušava da aproksimira vrijednost funkcije koja se modeluje, odnosno, čije ponašanje opisuju ulazni podaci.

Za razliku od nadgledanog učenja kod kog postoji jasna definicija relacije ulaza i izlaza, **nenadgledano učenje** (eng. *unsupervised learning*) predstavlja postupak obučavanja kod kog nisu poznate ove relacije. U tom slučaju, nije moguće definisati skup parova ulaz-izlaz, već se mreža obučava striktno nad nelabeliranim skupom. Nenadgledano učenje se najčešće koristi u data-majning (eng. *data mining*) primjenama.

Slično kao kod nadgledanog učenja, u nenadgledanom učenju se prepoznaju dvije osnovne podkategorije: klastering i asocijacija.

Klastering predstavlja grupisanje ulaznih podataka u klase koje nisu predefinisane od strane stručne osobe, gdje se način grupisanja (klasterisanja) najčešće sugeriše arhitekturom mreže (mada se u ovim primjenama rijetko koriste neuronske mreže). U slučaju klasteringa, mreža se obučava sa ciljem konvergencije ka prepoznavanju različitih klasa, do kojih se dolazi spontano za vrijeme obučavanja. Arhitekture mreža koje se koriste u ove svrhe su najčešće neke varijante Hopfildovih mreža, Bolcmanovih mašina ili samoorganizujućih mapa.

Asocijacija predstavlja pronalaženje međusobno povezanih varijabli u velikom skupu podataka i dominantno se koristi u data-majning aplikacijama, i vrlo rijetko uključuje korištenje neuronskih mreža. Neuronske mreže potencijalno podobne za asocijativno učenje su okidajuće mreže koje na neki način implementiraju Hebovo [107] pravilo (neuroni koji okidaju istovremeno međusobno se povezuju).

Pored pomenutih, postoje metode koje kombinuju nadgledano i nenadgledano učenje, gdje se koristi algoritamsko proširivanje ili generisanje skupova za obučavanje, mijenja proces obučavanja ili se on izvodi u različitim fazama. Ovi pristupi obučavanju, ukoliko se ne mogu podvesti pod neku od navedenih kategorija, najčešće se nazivaju **polunadgledano učenje** (eng. *semi-supervised learning*).

3.2.2 Varijante gradijentskog spuštanja

U praktičnim primjenama, gradijentsko spuštanje obično nije izvodljivo zbog činjenice da se izračunavanje vrši na fizičkim mašinama gdje postoji memorijsko ograničenje koje praktično onemogućava da se ažuriranje izvede odjednom nad cijelim skupom za obučavanje, a da, pri tom, obučavanje bude izvedeno u prihvatljivom vremenskom intervalu. Matematički gledano, ovakvo ažuriranje ima smisla, ali je u praksi rijetkost, osim u slučajevima veoma malih skupova podataka. Pored toga, gradijentsko spuštanje, formalno nazvano **grupno gradijentsko spuštanje** (eng. *Batch Gradient Descent*), ne dozvoljava aktivno (eng. *online*) ažuriranje, odnosno ažuriranje kako podaci dolaze. Iako u praksi često neupotrebljivo, grupno gradijentsko spuštanje garantuje konvergenciju ka globalnom minimumu za konveksne površine funkcije greške i konvergenciju ka lokalnom minimumu za nekonveksne površine.

Kako bi se razriješio ovaj problem, uvedeno je **stohastičko gradijentsko spuštanje** (eng. *Stochastic Gradient Descent*, SGD) koje ažurira težine za svaki element skupa za obučavanje ponaosob. Ova metoda je stohastička, jer se parovi ulaz-izlaz u procesu obučavanja biraju nasumično. Obično je praksa da se prođe kroz cijeli skup za obučavanje prije ponavljanja istih parova ulaz-izlaz pri obučavanju. Jedan korak ažuriranja, nad jednim parom u slučaju SGD, naziva se iteracija, a završetak jednog ciklusa iteracija nad svim elementima skupa za obučavanje naziva se epoha. U slučaju grupnog gradijentskog spuštanja, iteracija i epoha su ekvivalentni termini.

Stohastičko gradijentsko spuštanje se pokazuje kao računski efikasnije i vrlo upotrebljivo u slučajevima kada se obučavanje radi aktivno. S druge strane, SGD matematički ne garantuje konvergenciju, jer postoji mogućnost preskakanja minimuma, kako je objašnjeno

u poglavlju 2.6. U praksi se pokazuje da kada se definiše režim smanjivanja koeficijenta učenja, SGD može u većini slučajeva da konvergira u minimum. Ažuriranje parametara kod gradijenstkog spuštanja dato je jednačinom (2.49), dok je ažuriranje kod stohastičkog gradijentskog spuštanja dato jednačinom (3.1) gdje su gornjim indeksima i označeni indeksi parova ulaz-izlaz iz skupa za obučavanje nad kojima se radi ažuriranje.

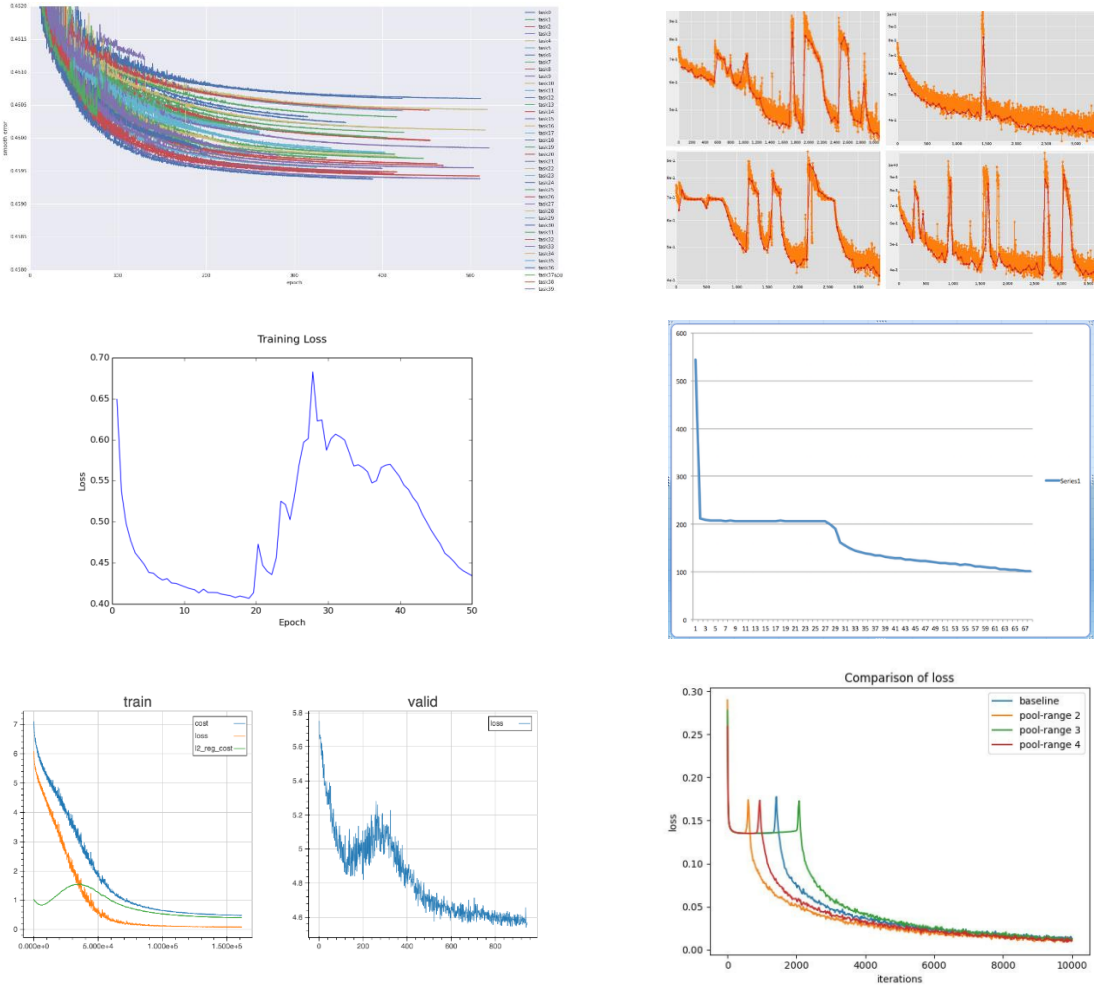
$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta E\left(\theta; x^{(i)}; y^{(i)}\right) \tag{3.1}$$

U praksi se, pored SGD pristupa koristi i takozvano **gradijentsko spuštanje sa mini-grupama** (eng. *Mini-Batch Gradient Descent*) gdje se ažuriranje izvodi odjednom na podgrupama nasumično odabranim iz skupa za obučavanje, i obično se pod SGD podrazumijeva ova vrsta gradijentskog spuštanja, budući da je SGD samo specijalna varijanta gradijentskog spuštanja sa mini-grupama, kod kog je veličina grupe jedan par ulaz-izlaz. Ovaj oblik gradijentskog spuštanja dat je jednačinom (3.2), gdje je n broj elemenata mini-grupe (grupe odabrane iz skupa za obučavanje za trenutnu iteraciju obučavanja).

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta E\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{3.2}$$

Razlog za ovakav pristup obučavanju je u tome da se smanji varijansa pri ažuriranju parametara i dovede do stabilnije konvergencije. Takođe, ovakav pristup pogodan je za optimizacije izračunavanja koje se mogu implementirati pri paralelnom izvršavanju na harveru namijenjenom za ovu svrhu. Ovaj pristup omogućava efikasno iskorištenje procesorskih jezgara i memorije.

S druge strane, sve varijante stohastičkog gradijentskog spuštanja imaju problem sa fluktuacijama funkcije gubitka za vrijeme obučavanja mreže, pa je funkcija njene vrijednosti u odnosu na vrijeme često vrlo nelinearna. Slika 3.1 sadrži neke zanimljive primjere ponašanja funkcije gubitka u odnosu na broj iteracije.



Slika 3.1 [108] – *Primjeri ponašanja funkcija gubitka kod različitih vrsta optimizatora. Ovi primjeri preuzeti su iz kolekcije neobičnih grafika funkcije gubitka i prikazani su radi pojašnjenja mogućeg ponašanja vrijednosti funkcije gubitka u odnosu na vrijeme. Specifični parametri optimizatora nisu poznati ni relevantni.*

Izbor veličine mini-grupe (ili grupe, budući da se termin grupa češće koristi) obično je u opsegu od 50 do 256, zavisno od hardverske platforme i načina obučavanja, mada nisu rijetke varijante sa većim i manjim grupama. Odabrana veličina grupe može značajno da utiče na konvergenciju i ponekad povećavanje grupe može da riješi problem nestajućih gradijenata.

3.2.3 Učenje za više zadataka

Kako se moglo naslutiti iz Slike 3.1, moguće je da se nad jednom neuronskom mrežom definišu različite funkcije gubitka, jer mreža ne mora nužno da ima jedan izlaz, niti mora da na isti način optimizuje sve svoje izlaze, pa se tako, u specifičnim slučajevima, nad mrežom definiše više funkcija gubitaka.

Korištenje više različitih funkcija gubitaka je fundamentalno posljedica pristupa (odnosno, pojavno svojstvo) u obučavanju mreže, više nego metoda obučavanja. Razlog

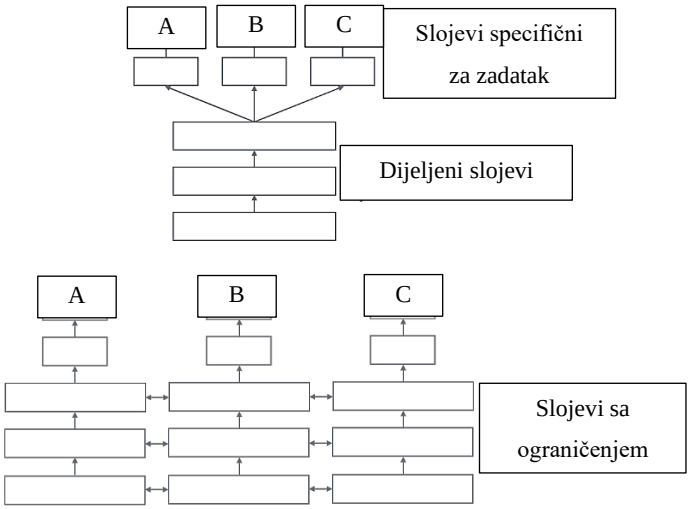
postojanja više funkcija gubitaka je činjenica da se mreža istovremeno obučava za rješavanje više sličnih zadataka ili za više zadataka koji dijele neku komponentu (neki element datih zadataka je sličan). Ovaj pristup obučavanju naziva se učenje za više zadataka (eng. *Multi-Task Learning*, MTL).

Učenje za više zadataka iskorišteno je u velikom broju praktičnih primjena, uključujući obradu jezika [109], prepoznavanje govora [110], računarski vid [111] i pronalaženje lijekova [112]. Učenje za više zadataka često se imenuje i alternativnim nazivima, kao što su združeno učenje (eng. *joint learning*), učenje učenja (eng. *learning to learn*) i učenje pomoćnim zadacima (eng. *learning with auxiliary tasks*). Suštinski, MTL poboljšava generalizaciju iskorištavajući domensko znanje iz oblasti drugih vezanih ili sličnih zadataka [113].

U mašinskom učenju, termin **induktivna pristrasnost** odnosi se na bilo kakvu modifikaciju mreže koja uzrokuje da model preferira neki skup hipoteza u odnosu na druge. Na primjer, L1 regularizacija je oblik induktivne pristrasnosti ka odabiranju rijetkih koeficijenata. Slično, konstrukcijom mreže na takav način da se optimizuje za vezane zadatke, korištenjem induktivnog transfera, odnosno prebacivanja induktivne pristrasnosti sa jednog zadatka na drugi, mreža se podešava tako da preferira i hipoteze iz vezanog zadatka i na taj način generalizuje razumijevanje na objašnjenje oba zadatka umjesto samo traženog. Ovo omogućava mreži da uči domensko znanje iz vezane oblasti i na taj način formira generalizovano razumijevanje ciljne oblasti, kroz indukovani domenski transfer.

Tipično, MTL se postiže na jedan od dva glavna načina: tvrdo i meko dijeljenje parametara (eng. *parameter sharing*). Češće korišteno, tvrdo dijeljenje parametara podrazumijeva da oba zadatka koriste iste skrivene slojeve na koje su nadovezani dodatni skriveni slojevi specifični za pojedinačne zadatke. Ovaj pristup MTL obučavanju značajno povećava generalizaciju i pokazuje se da smanjuje vjerovatnoću overfitovanja dijeljenih parametara n puta, gdje je n broj vezanih zadataka [114] [115].

Meko dijeljenje parametara podrazumijeva da svaki od zadataka ima svoj odvojen model (neuronsku mrežu), ali se pri obučavanju parametri iz odabranih slojeva zajedno regularizuju. Odnosno, regularizacija se vrši tako da se parametri iz odabranih slojeva podstiču na formiranje sličnih vrijednosti. Slika 3.2 ilustruje razliku između tvrdog i mekog dijeljenja parametara [116] [117].



Slika 3.2 [113] – *Tvrdo (gore) i meko (dole) dijeljenje parametara*

Suštinski, postoji nekoliko glavnih razloga zašto MTL poboljšava generalizaciju. Prije svega, MTL implicitno povećava skup podataka za obučavanje, jer se dijeljeni slojevi obučavaju na većem broju parova ulaz-izlaz, odnosno na svim parovima iz skupa za obučavanje za svaki od vezanih zadataka. Pored toga, kako svaki skup podataka (odnosno svaki domen problema) karakteriše određen obrazac šuma, mreža uči da ignoriše šum iz svih domena i na ovaj način bolje generalizuje nad svim zadacima. Slično, MTL omogućava mreži da se fokusira na one elemente iz svih skupova za obučavanje (odnosno, iz svih domena) koji su relevantni, zanemarujući one koji postoje samo u jednom.

Dodatno, ukoliko postoje dva zadatka, A i B koji dijele neko obilježje, moguće je da je isto obilježje lakše naučiti obučavanjem mreže za zadatak A u odnosu na obučavanje za zadatak B, iako je potencijalno ciljni zadatak mreže zadatak B. Ovaj koncept u obučavanju naziva se prisluškivanje (eng. *eavesdropping*) [118]. Pored toga, MTL mreži dodaje pristrasnost ka preferiranju reprezentacija koje i ostali zadaci preferiraju, pa se pokazuje da mreža koja dobro generalizuje za više sličnih zadataka, dobro generalizuje i kada se primjenjuje na novim zadacima iz slične okoline [119]. Pored navedenog, MTL djeluje i kao regularizatorski mehanizam, smanjujući rizik od overfitovanja i smanjujući Rademaherovu kompleksnost modela [120] (odnosno, sposobnost mreže da fituje nasumični šum).

3.3 NESTAJANJE I EKSPLOZIJA GRADIJENATA

Kako broj slojeva u neuronskoj mreži raste, povećava se broj parametara koje mreža može da nauči i povećava se dubina mreže po kojoj SGD unazad propagira gradijente. Kada se gradijenti propagiraju unazad sloj po sloj, ukoliko na slojevima postoji funkcija koja ih pri

propagaciji umanjuje ili uvećava, numeričke vrijednosti gradijenata kao rezultat propagacije unazad kroz slojeve sa ovakvom funkcjiom mogu eksponencijalno da se smanjuju ili povećavaju, s obzirom na broj slojeva.

U slučaju da su gradijenti propagirani unazad kroz sloj čije su težine veza male i gdje postoji aktivaciona funkcija čiji izvod dodatno smanjuje vrijednosti gradijenata, vrijednosti propagiranih gradijenata mogu da postanu značajno male. U slučaju kada te vrijednosti postanu toliko male da učine da se mreža ne može obučavati, ova pojava se naziva problem nestajućeg gradijenta (eng. *vanishing gradient problem*). Problem nestajućeg gradijenta u praksi može da se manifestuje na dva načina: smanjenje vrijednosti gradijenata do mjere da se na mreži parametri više ne ažuriraju (obično na početnim slojevima, jer su na njima gradijenti najmanji) ili smanjenje gradijenata do mjere kada se ne mogu čuvati kao podaci sa pokretnim zarezom, gdje dolazi do aritmetičkog potkoračenja (eng. *underflow*) i pojavljivanja NaN (eng. *Not a Number*) vrijednosti, gdje se cijela implementacija mreže ruši kao rezultat ažuriranja parametara propagiranim NaN vrijednostima. Mreže narušene NaN vrijednostima neophodno je ponovo obučavati.

S druge strane, ukoliko izvodi aktivacionih funkcija i postojeće vrijednosti težina favorizuju uvećanje gradijenata, sličan problem može da se javi. U slučaju kada propagirane vrijednosti postanu toliko velike da izazovu divergenciju mreže, ovaj problem naziva se problem eksplodirajućeg gradijenta (eng. *exploding gradient problem*). Problem eksplodirajućeg gradijenta obično se pojavljuje u slučajevima kada je koeficijent učenja previsok za datu mrežu i problem koji se rješava, pa se ažuriranja dešavaju u prevelikim koracima. Ovaj problem se često može korigovati odgovarajućim podešavanjem koeficijenta učenja i veličine grupe. Pored eksplodirajućeg gradijenta, eksponencijalno povećavajuće vrijednosti mogu se pojaviti i pri inferenciji i obično su uzrokovane istim mehanizmom koji dovodi do eksplodirajućeg gradijenta. Aktivacione funkcije koje preslikavaju u otvoren interval često dovode do eksplozije vrijednosti pri inferenciji, dok uspješno rješavaju problem nestajućeg gradijenta. Eksplodirajući gradijenti, slično kao nestajući gradijenti, mogu da izazovu NaN vrijednosti, pri čemu su u ovom slučaju one izazvane aritmetičkim prekoračenjem (eng. *overflow*). Odsijecanje gradijenata (eng. *gradient clipping*) je jedan od načina rješavanja problema eksplodirajućih gradijenata, gdje se vrši saturacija pri dodavanju vrijednosti gradijenata, odnosno gradijenti se ograničavaju na neki predefinisani konačni interval korištenjem neke odabrane norme [121]. Kod nestajućih gradijenata, ovo nije izvedivo, jer i u slučaju da se niske vrijednosti postave na nulu, neuroni do kojih se propagiraju nikada neće biti

ažurirani i na taj način će biti onemogućeno dalje obučavanje. Dodatno, najčešće su individualni elementi ulaznog tenzora i izlaznog tenzora na svakom sloju normalizovani na opseg $[0, 1]$ ili $[-1, 1]$, radi izbjegavanja problema sa eksplozijom gradijenata, te izbjegavanja unošenja dodatnih koeficijenata u aktivacione funkcije.

Nestajući gradijenti, kako je već rečeno, mogu se riješiti uvođenjem odgovarajućih aktivacionih funkcija. Konkretnije, najčešće se, radi izbjegavanja ovog problema, u skrivenim slojevima mreža koristi ReLU aktivaciona funkcija ili neka od njenih varijanti. Uz to, izbjegava se korištenje sigmoidne funkcije i hiperbolnog tangensa za aktivaciju u skrivenim slojevima, osim ako ograničavanje na zatvoren interval nije neophodno. Pored toga, zavisno od optimizatora i biblioteke koja se koristi za implementaciju neuronske mreže, promjena veličine grupe može značajno da pomogne izbjegavanju ovog problema. Konkretnije, povećavanje veličine grupe u većini biblioteka za rad sa neuronskim mrežama smanjiće mogućnost pojavljivanja nestajućih (i eksplodirajućih) gradijenata. Dodatno, efikasan način rješavanja nestajućih gradijenata je i korištenje preskačućih veza, kako je navedeno u poglavlju 2.7, kao i korištenje odgovarajućih mehanizama inicijalizacije, kako je opisano u poglavlju 3.7.

Predložena rješenja primjenjiva su kod većine arhitektura neuronskih mreža, no u slučaju rekurentnih neuronskih mreža, dodatna mogućnost je korištenje LSTM jedinica umjesto standardnih rekurentnih neurona. Korištenje LSTM jedinica izbjegava pojavljivanje nestajućih gradijenata pri propagaciji na prošla stanja.

Još jedan način izbjegavanja oba problema je korištenje odgovarajuće regularizacije pri procesu obučavanja. Kako je ranije rečeno, regularizacija, pored toga što pomaže generalizaciji, umanjuje vjerovatnoću pojavljivanja nestajućih gradijenata. Slično tome, korištenje MTL obučavanja smanjuje vjerovatnoću pojavljivanja ovih problema, sa povećanjem broja vezanih zadataka.

Pojava nestajućih gradijenata se pri obučavanju manifestuje sporom promjenom funkcije gubitka u odnosu na broj iteracija, nestabilnošću modela i velikim skokovima u funkciji gubitka za vrijeme obučavanja i pojavom NaN vrijednosti. Pojava eksplodirajućih gradijenata manifestuje se brzim povećavanjem vrijednosti parametara (težina) mreže za vrijeme obučavanja, pojavom NaN vrijednosti u parametrima mreže i pojavom vrijednosti gradijenata iznad 1 na svim neuronima svih slojeva mreže [122] [123].

3.4 SPREČAVANJE OVERFITINGA I GENERALIZACIJA

Kako je pomenuto u poglavlju 2.7, generalizacija je jedan od ključnih ciljeva pri konstrukciji i obučavanju neuronskih mreža, pogotovo u slučaju kada je cilj praktična primjena u domenu problema, gdje je broj uzoraka doveden na ulaz potencijalno redove veličine brojniji od uzoraka datih skupom za obučavanje. U istom poglavlju je predstavljena regularizacija kao vrlo bitna metoda za sprečavanje overfitinga i omogućavanje generalizacije. Pored toga, u poglavlju 3.2 predstavljen je pristup učenju za više zadataka, odnosno MTL. Ovdje su predstavljeni ostali, često korišteni, pristupi [53] [71] [113].

Prvi, i očigledan, pristup je povećavanje broja primjera za obučavanje. Ovo se može izvesti dodatnim labeliranjem ulaza ili algoritamskim proširivanjem ulaznog skupa. Zavisno od domena problema, ulazni skup može da se proširi na različite načine. Na primjer, u slučaju da se na ulaz mreže dovode slike čija orijentacija može biti proizvoljna (kakav je slučaj sa aero-snimcima [1]), tada se skup može proširiti algoritamskim generisanjem dodatnih rotiranih i reflektovanih slika za svaki od ulaza. Pored toga, moguće je dodavati šum ili mijenjati osobine osvjetljenja ulaznih slika, kontrast, boju, te vršiti razne vrste korekcija i alteracija nad ulaznim slikama. Ekvivalentne operacije mogu se uraditi i sa zvukom ili bilo kojom drugom vrstom ulaza za koji postoji domensko znanje i smislena transformacija. Naravno, postoje i sofisticiranije metode proširivanja ulaznog skupa podataka, kao što je konstrukcija generativnog modela ili modela za transformaciju ulaza, kao što je, na primjer, GAN arhitektura.

Ukoliko model sadrži relativno mali broj parametara i ukoliko postoje odgovarajući alati (kao što je, na primjer, direktna evaluacija u Tensorflow [124] alatu), moguće je ručno pregledati naučena obilježja i odbacivati ona koja za datu mrežu nemaju smisla. Ovo je ekvivalentno debagovanju računarskog koda, pri čemu se debaguje konkretan model nakon određenog broja epoha.

Analogno regularizaciji, **odbacivanje** (eng. *dropout*) je metoda kod koje se pri ažuriranju parametara na osnovu gradijenata nasumično odbacuje određeni procenat neurona (po sloju). Ova metoda praktično odstranjuje efekat odbačenih neurona i rezultuje obučavanjem preostale mreže koja je znatno jednostavnija. Na ovaj način, sprečava se specijalizacija različitih neurona za istu svrhu. Ovaj pristup pokazuje se kao vrlo efikasan način regularizacije. Naravno, odbacivanje se radi samo pri obučavanju. Pri inferenciji, svi neuroni mreže ostaju aktivni.

Relativno jednostavan način sprečavanja overfitinga je i **rano zaustavljanje** (eng. *early stopping*). Rano zaustavljanje je prekidanje obučavanja prije konvergencije nad skupom za obučavanje, kada se uoči da se ne pojavljuje napredak na validacionom skupu. Ovo se pokazuje kao vrlo efikasno, ukoliko je validacioni skup pogodno sastavljen i ukoliko se rekonstrukcija mreže ne radi previše često. U slučaju da se rekonstrukcija mreže radi svaki put kada se uoče nezadovoljavajuće performanse nad validacionim skupom, efektivno se može desiti overfitovanje na nivou konstrukcije modela, gdje rekonstruisana arhitektura modela preferira izabrani validacioni skup. Dakle, u idealnom slučaju, proces obučavanja bi trebalo da minimizira broj poređenja sa validacionim skupom.

Ranije je napomenuto da se podaci iz skupa za obučavanje unutar svake epohe mreži daju nasumično. Drugim riječima, u svakoj epohi podaci za obučavanje će dolaziti različitim redoslijedom. Ovo poboljšava generalizaciju, jer se mreža ne obučava da preferira određene uzorke. S druge strane, nekad je poželjno da se podaci na ulaz dovode u nekom predefinisanom redoslijedu, ukoliko taj redoslijed može da pomogne konvergenciji mreže ili generalizaciji. Metoda određivanja ovog redoslijeda naziva se **obučavanje po kurikulumu** (eng. *curriculum learning*) [125]. Primjer za ovo je sortiranje ulaznih primjera prema njihovoj težini učenja, gdje je težina učenja subjektivno evaluirana od strane obučenog labelara [126].

Kada se parametri neuronskih mreža prije početka obučavanja inicijalizuju sa jediničnom varijansom i srednjom vrijednošću, za vrijeme obučavanja ova normalizacija se gubi, jer se parametri ažuriraju na različite načine. **Grupna normalizacija** (eng. *batch normalization*) [127] je proces koji renormalizuje aktivacije u svakoj iteraciji, odnosno nakon svakog ažuriranja mini-grupe. Dodatno, grupna normalizacija djeluje kao regularizator i ponekad eliminiše potrebu za odbacivanjem.

Pored normalizacije, dodavanje Gausovog šuma gradijentima se pokazuje kao efikasan način izbacivanja mreže iz lokalnih minimuma i korekcije loše inicijalizacije parametara. Ovo je posebno korisna metoda kada se radi sa dubljim arhitekturama [128].

3.5 ARHITEKTURE MREŽA

Sa programskog stanovišta, sloj je skup operacija koji prihvata tenzor na ulazu i generiše tenzor na izlazu, a njegovi interni parametri mogu da se ažuriraju na osnovu gradijenata dovedenih na izlaz, pri čemu se podrazumijeva propagacija unazad. Drugim riječima i manje formalno, sloj je skup operacija i parametara neuronske mreže koji se ponaša kao najmanji i

atomični dio mreže. Naravno, ova slobodna definicija zavisi od konkretne biblioteke koja se koristi, ali je dovoljno koncizna za svrhe enumeracije.

Iz prethodno rečenog, može se naslutiti da se savremene arhitekture neuronskih mreža uglavnom sastoje od međusobno povezanih slojeva. Ove slojevite arhitekture su najrelevantnije za temu ove disertacije, i shodno tome se nameće potreba za enumeracijom najčešće korištenih tipova slojeva i njihovom praktičnom upotrebom.

Konkretna implementacija slojeva i klasifikacija toga šta može da se posmatra kao sloj obično zavisi od konkretne biblioteke ili frejmvorka (eng. *framework*) koji se koristi za implementaciju mreže. Iako je matematički model neuronskih mreža isti, različite biblioteke različito modeluju mreže na programskom nivou. Ovo znači da neke biblioteke određene elemente matematičkog modela programski modeluju kao dio sloja, dok druge iste elemente razdvajaju. Suštinski, ovo ne pravi značajne razlike u objašnjenju, i ovdje će slojevi biti prezentovani bez preference ka nekoj specifičnoj biblioteci.

Prije svega, vrijedi napomenuti da se aktivacione funkcije, zavisno od biblioteke, nekada specifikuju na nivou sloja, a nekada izdvajaju u odvojen sloj. Bilo da se radi o odvojenom sloju za aktivacionu funkciju ili aktivacionoj funkciji ugrađenoj u sloj, parametri aktivacione funkcije su skoro uvijek oni definisani matematičkim modelom.

Slična separacija programskog modela postoji i kod funkcija gubitka. Većina alata i frejmvorka funkcije gubitka predstavlja slojevima, a ne odvojenim entitetima. Dodatno, ne dozvoljavaju sve biblioteke rad sa više funkcija gubitka, te neke definišu funkciju gubitka na nivou posljednjeg sloja. Slično kako je bio slučaj sa aktivacionim funkcijama, funkcije gubitka mogu, a ne moraju da se posmatraju kao slojevi.

Suštinski, za svaku aktivacionu funkciju i za svaku funkciju gubitka može se smatrati da postoji istoimeni sloj. Optimizatorski mehanizmi se obično specifikuju na nivou cijele mreže, mada neke biblioteke dozvoljavaju i specifikaciju različitih optimizatora za različite funkcije gubitka.

Suštinski, tenzor koji se dovede na ulaz neuronske mreže propagira se kroz sve slojeve mreže po specifikovanim rutama između slojeva. Principijelno, ove rute mogu biti bilo kakve, ali većina biblioteka zahtijeva da slojevi i njihovo međusobno povezivanje čine usmjeren aciklični graf (eng. *Directed Acyclic Graph*, DAG), odnosno onemogućavaju postojanje ciklusa na nivou povezivanja slojeva (većina biblioteka sadrži rekurentne i LSTM slojeve koji

interno sadrže cikluse). Svaki tip sloja, zavisno od skupa operacija koje predstavlja, može biti konfigurisan određenim, njemu svojstvenim, hiperparametrima. Kako je ranije rečeno, izbor slojeva i njihovih parametara i njihovo međusobno povezivanje predstavljaju arhitekturu neuronske mreže.

3.5.1 Tipovi slojeva

Radi predstavljanja uopštenije slike, ovdje su razvrstani najčešće korišteni slojevi i njihovi glavni parametri. Naravno, parametrizabilnost slojeva u praktičnim slučajevima zavisi od konkretne biblioteke koja se koristi, ali na nešto apstraktnijem nivou, svi izloženi tipovi slojeva moraju da imaju implementaciju prikazanih parametara, bez obzira na biblioteku [51] [52] [129] [130].

Prije svega, većina slojeva koji se koriste dijeli određen skup parametara, pa će oni ovdje biti navedeni odvojeno. Na nivou izlaza slojeva može da se definiše nekoliko operacija. Jedna ovakva operacija je aktivacija, pa se, u tom slučaju, definiše aktivaciona funkcija nad izlazima sloja. Pored aktivacije, na svakom sloju može se odrediti način regularizacije izlaza, sa parametrima regularizacije, normalizacija izlaza (odnosno skaliranje na interval), ograničavanje (odsijecanje ili saturacija) vrijednosti izlaza, kao i gradijentsko odsijecanje.

Osnovni sloj kog skoro sve mreže za klasifikaciju sadrže je **potpuno povezani sloj**. Ovaj tip sloja predstavlja sloj klasične FFNN mreže i suštinski se sastoji od matrice težina i prednaponskog vektora. Postojanje prednaponskog vektora se nekad izuzima, pa je indikator njegovog postojanja parametar potpuno povezanog sloja. Naravno, za svaki potpuno povezani sloj specifikuje se broj neurona u njemu.

Radi konstrukcije kompleksnijih arhitektura, obično se koriste **slojevi za rutiranje**. Ova vrsta slojeva uključuje slojeve za promjenu redoslijeda ulaznih vrijednosti ili promjenu prostornog rasporeda ili dimenzionalnosti (na primjer, sloj za izravnavanje sve elemente tenzora projektuje na elemente vektora). Parametri ovih slojeva specifikuju način promjene dimenzionalnosti ili prostornog rasporeda. U klasu slojeva za rutiranje spadaju i slojevi za podjelu (odnosno kopiranje vrijednosti), slojevi za sječenje (odnosno razdvajanje ulaza na više izlaza po nekoj osi ili kriterijumu) i slojevi za konkatenaciju. Ovi slojevi, izuzev sloja za dijeljenje, samo prosljeđuju vrijednosti sa ulaza i propagiraju gradijente unazad, bez promjena. Sloj za podjelu propagira gradijente unazad ili sumiranjem ili prosječnom vrijednošću, jer svi kopirani izlazi doprinose izmjenama.

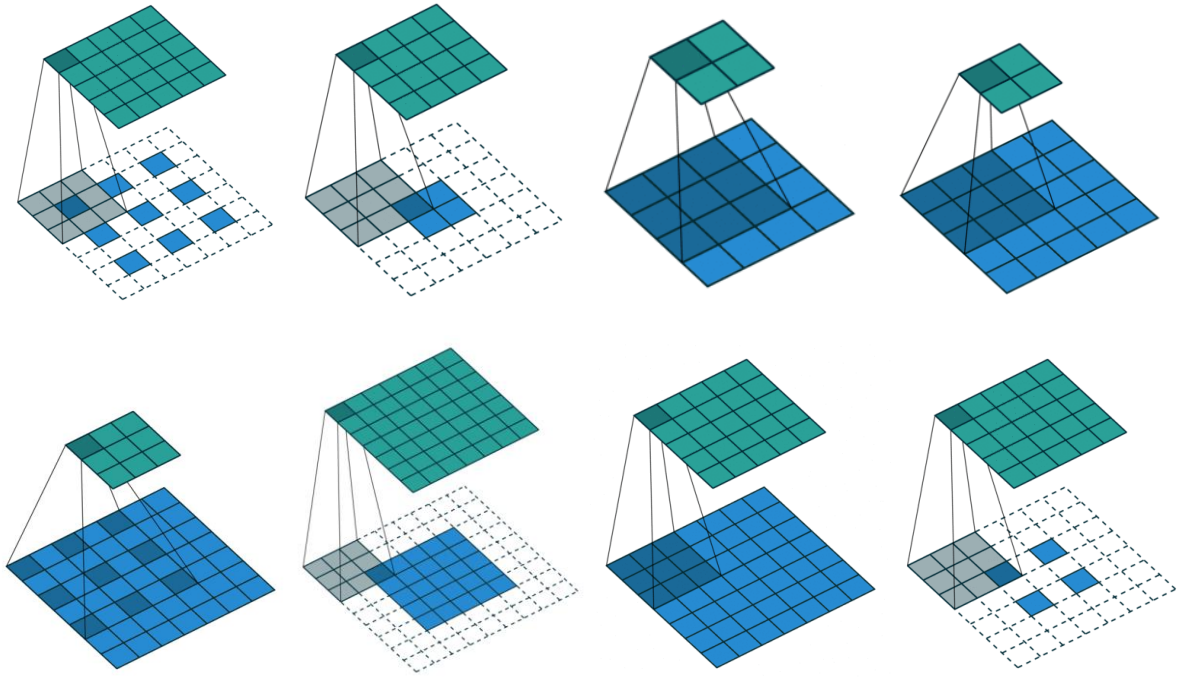
Pored slojeva za rutiranje, u sličnu svrhu se često koriste i **slojevi sa osnovnim matematičkim operacijama**, kao što su sabiranje, oduzimanje, množenje, dijeljenje, stepenovanje i slično, koji prihvataju dva ulaza iste dimenzionalnosti i izvode nad odgovarajućim parovima istu matematičku operaciju.

Konvolucioni slojevi, za razliku od potpuno povezanih, podešavaju se sa većim brojem hiperparametara. Ovi slojevi, kako je rečeno u poglavlju 2.2, izvode operaciju konvolucije nad ulaznim tenzorom sa svakim od definisanih kernela. Tipično, ovi slojevi sadrže parametre broja, veličine i dimenzionalnosti kernela, kao i načina izvođenja konvolucije (gdje se konvolucija može raditi odvojeno po kanalima ili spojeno, kako je diskutovano ranije). Pored veličine i dimenzionalnosti, vrlo često se koristi i parametar koraka, koji određuje za koliko elemenata ulaza se, po svakoj osi ulaza, pomjera klizeći prozor diskretne matrične konvolucije. Ovaj parametar igra značajnu ulogu u konstrukciji mreže, jer određuje veličinu izlaza. Povećavanjem veličine kernela i koraka prozora smanjuje se veličina izlazne slike. Parametar koji se koristi za kompenziranje ove promjene se naziva parametar popunjavanja i specifikuje način popunjavanja bočnih elemenata u slučaju da se specifikuje da izlaz mora biti istih dimenzija kao i ulaz. Dodatno, i konvolucioni slojevi mogu da uključuju prednaponski član.

Pored konvolucionih slojeva, često se koriste i **slojevi za transponovanu konvoluciju** (nekad pogrešno nazvanu dekonvolucijom) koji se koriste kako bi se aproksimirala operacija inverzna onoj izvedenoj od strane konvolucionih slojeva. Budući da stvarna dekonvolucija nije praktično izvediva, zbog gubitka dijela informacija pri konvoluciji, transponovana konvolucija obično podrazumijeva i neku vrstu umetanja novih podataka, dodavanja šuma ili kopiranja elemenata ulaza. Ova vrsta sloja koristi se kod autoenkodera i generatorskih modela koji ciljaju da povećavaju dimenzije ulaza. S obzirom na to da i slojevi za transponovanu konvoluciju i slojevi za konvoluciju često sadrže parametar za dilaciju (nadodmjeravanje istog elementa ili ekstrapoliranje međuelemenata s ciljem povećanja veličine izlaza; dodavanje novih elemenata u tenzor, između onih produkovanih operacijom konvolucije), većina operacija koje se mogu implementirati slojevima za transponovanu konvoluciju mogu se implementirati i slojevima za konvoluciju.

Slojevi za agregaciju (eng. *pooling layer*) i **slojevi za nadodmjeravanje** (eng. *upsampling layer*) služe za smanjivanje i povećavanje dimenzija ulaznih podataka, respektivno. Operacija pododmjeravanja je diskutovana ranije, i obično uključuje agregaciju po maksimumu i prosječnoj vrijednosti. Operacija nadodmjeravanja obično sekvencijalno

ponavlja podatke sa ulaza unutar zadate veličine prozora. Ovi slojevi, slično kao i slojevi za konvoluciju, sadrže parametre veličine kernela i koraka prozora na osnovu kojih se određuje veličina ulazne slike i priroda smanjenja ili uvećanja ulaza. Slika 3.3 ilustruje nekoliko operacija koje uključuju klizeći prozor i parametar koraka, uključujući i konvoluciju i transponovanu konvoluciju.



Slika 3.3 [131] – *Transponovana konvolucija sa korakom 2 i popunjavanjem, transponovana konvolucija sa korakom 1 i bez popunjavanja, konvolucija sa korakom 1, konvolucija sa korakom 2, konvolucija sa dilacijom, konvolucija sa korakom 1 i popunjavanjem, transponovana konvolucija sa korakom 1 bez popunjavanja, transponovana konvolucija sa korakom 2 bez popunjavanja (redom slijeva udesno, odozgo ka dole; tok podataka odozdo ka gore)*

Pored navedenih tipova slojeva, vrlo često se koriste i rekurentni slojevi, koji uključuju RNN neurone, LSTM i GRU neurone. Ovi slojevi neće biti razmatrani, jer njihovo uvođenje mijenja prirodu modela na rekurentni i time mijenja domen primjene neuronske mreže.

U fazi obučavanja, postoje i pomoćni slojevi od kojih se često koriste slojevi za odbacivanje (eng. *droput layers*) i slojevi za prikazivanje međuvrijednosti (slojevi za debagovanje). Ovi slojevi se koriste za automatsko ili manuelno indukovanje pristrasnosti mreže, regularizaciju i generalno posmatranje ponašanja mreže i njenih performansi.

Suštinski, najveći broj neuronskih mreža sačinjen je samo od navedenih slojeva, ali većina biblioteka za rad sa neuronskim mrežama dozvoljava mogućnost manuelne

implementacije bilo koje vrste sloja. Ovdje je bitno napomenuti da se, iako ovi modeli slojeva variraju od biblioteke do biblioteke, ručno specifikovani modeli obično programski modeluju kroz funkciju za propagaciju ulaza na izlaze i funkciju propagacije gradijenata sa izlaza na ulaze. Ove funkcije za propagaciju ne moraju poštovati predstavljeni matematički model i mogu da sadrže pravila za ažuriranje parametara i izračunavanje gradijenata potpuno nekonzistentna sa prezentovanim matematičkim modelom. Naravno, slučajevi kada se ovakvo nešto koristi su izuzetno rijetki i generalno se cilja da propagacija unazad bude matematički konzistentna u odnosu na propagaciju unaprijed.

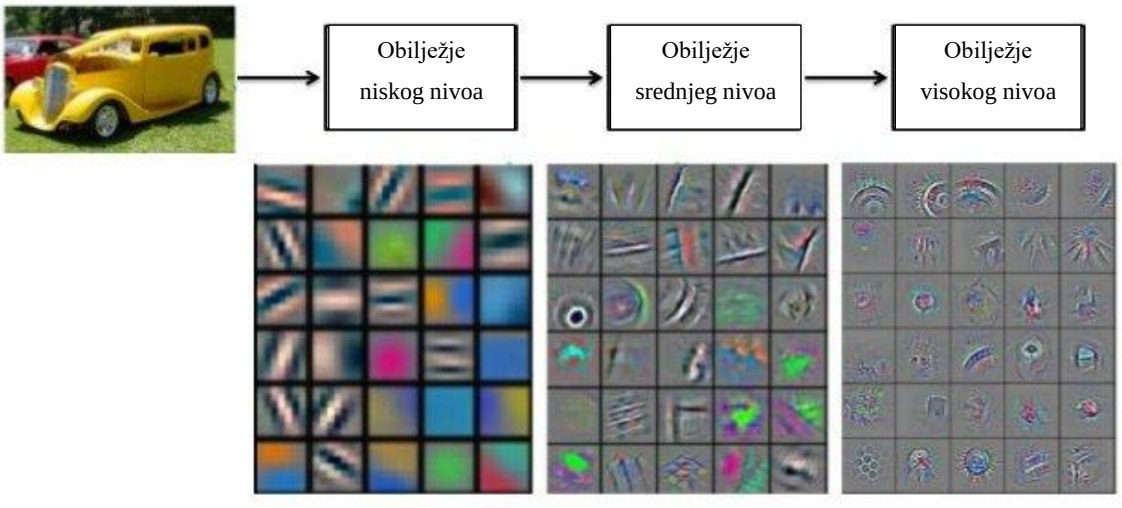
3.5.2 Specifične konstrukcije

Iako je rečeno da neuronske mreže sačinjene od slojeva mogu u praksi da čine bilo kakvu arhitekturu, te da, pored generalizovane klasifikacije date u poglavlju 2.3, ne postoji ustanovljen način klasifikacije neuronskih mreža, neki oblici konstrukcije, ili arhitekturalni principi, izdvajaju se kao posebni i od posebnog značaja pri konstrukciji specijalizovanih arhitektura.

Prije svega, prvobitna arhitektura „mreže u mreži“ [132], koja je inspirisala Inception arhitekturu koju je Google koristio za prevazilaženje ljudske tačnosti u klasifikaciji slika opšte namjene, uvela je ideju proširivanja kompleksnosti operacija individualnim slojevima, gdje se, umjesto korištenja konvolucionog kernela, cijeli kernel mijenjao ugniježdenom neuronskom mrežom. Ova ideja inspirisala je veliki broj dubokih arhitektura i pokazala se kao ključ za savladavanje ImageNet skupa.

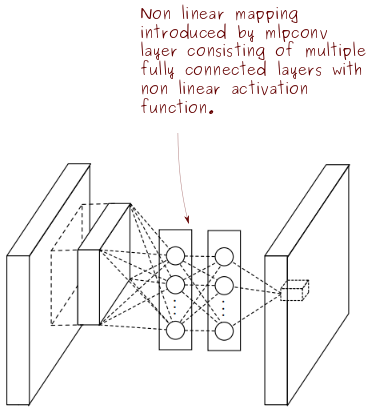
Pristup gniježdenja mreža sastojao se od dva elementa: uvođenja MLPconv operacije (odnosno zamjene linearnih filtara na konvolucionim slojevima sa multilinearnim perceptronima) i globalnog prosječnog pododmjeravanja (eng. *Global Average Pooling*) koje je uklonilo potpuno povezane slojeve na izlazima, mijenjajući ih konvolucionim slojem sa brojem kernela koji je odgovarao broju klasa, gdje su se izlazi određivali na osnovu prosječne vrijednosti izlaza konvolucije svakog od kernela posljednjeg sloja.

Prije uvođenja multilinearnih perceptrona, mreže su koristile klasične konvolucione linearne filtre, gdje su kompleksne reprezentacije obilježja učene postepeno kombinujući rezultate prethodnih slojeva, kako je ilustrovano Slikom 3.4.



Slika 3.4 [133] – *Progresivno učenje kompleksnijih obilježja, sloj po sloj*

Multilinearni perceptroni omogućavaju učenje kompleksnijih obilježja na nivou jednog sloja, zamjenjujući konvolucioni kernel ugniježdenom višeslojnom neuronskom mrežom (odatle i naziv multilinearni perceptron). Slika 3.5 prikazuje šemu multilinearnog perceptrona, odnosno MLPconv operacije, iz originalnog rada „mreža u mreži“.



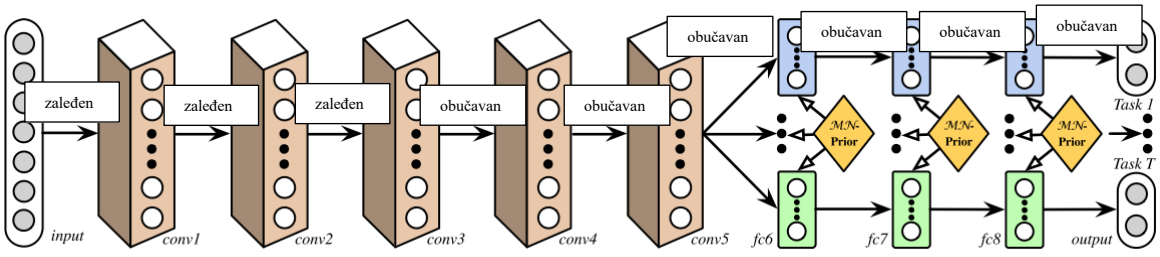
Slika 3.5 [132] – *Multilinearni perceptron (MLPConv operacija), slika iz originalnog rada*

Klasično, konvolucione mreže završavaju potpuno povezanim slojevima koji imaju ulogu klasifikatora koji klasifikaciju vrši na osnovu obilježja naučenih od strane prethodnih (obično konvolucionih) slojeva mreže. Arhitektura predložena kao „mreža u mreži“ ovaj oblik konstrukcije klasifikatora zamjenjuje uvođenjem nove operacije agregacije, pomenutom globalnom prosječnom agregacijom. Ova operacija implementirana je tako da posljednji MLPconv sloj na izlazu sadrži onoliko kanala koliko postoji klasa, a vrijednosti vjerovatnoća za svaku od klasa dobijaju se računanjem prosječne vrijednosti za sve izlaze unutar jednog kanala. Prednosti ovog pristupa su u smanjenju broja parametara za obučavanje i činjenici da

globalna prosječna agregacija sumira cijele izlaze posljednjeg sloja čineći mrežu robusnijom na prostornu translaciju ulaza.

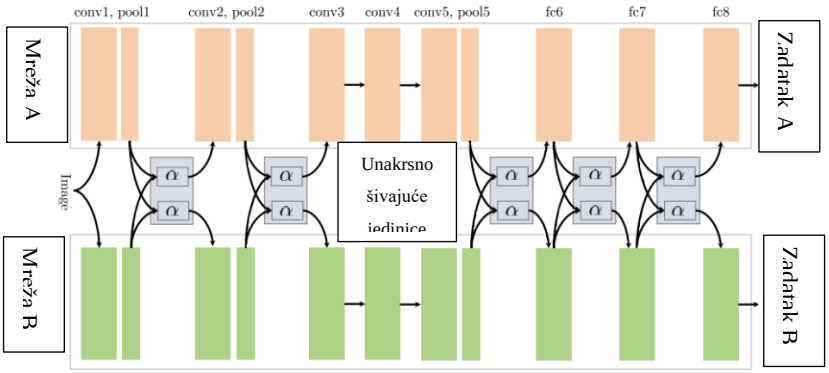
Pored navedene ugniježdene arhitekture, često se javlja potreba za specijalizacijom arhitekture s obzirom na MTL pristup obučavanju. Ovaj pristup obično podrazumijeva neku vrstu grananja mreže ili posebne arhitekturalne adaptacije.

Mreže sa dubokim vezama (eng. *deep relationship network*) [134], umjesto dijeljenja parametara samo na konvolucionim slojevima, koriste posebne slojeve za povezivanje između individualnih zadataka kako bi pomogle učenju međusobnih veza između zadataka, kako je ilustrovano Slikom 3.6.



Slika 3.6 [113] – *Mreža sa dubokim vezama*

Uvođenjem unakrsno-šivajućih jedinica (eng. *cross-stitch unit*) [135], unakrsno-šivajuće mreže koriste linearnu kombinaciju izlaza jedne mreže za učenje zadataka u drugoj, pri čemu među-šivajuće jedinice povezuju dvije odvojene arhitekture mreža, kako ilustruje Slika 3.7.



Slika 3.7 [113] – *Među-šivajuća mreža*

Pored pomenutih mreža za MTL, postoje mnoge druge arhitekture, koje iskorištavaju slične arhitekturalne koncepte pri povezivanju. Ovdje su izložene pomenute arhitekture, jer dobro ilustruju indukovanje pristrasnosti mreže ka rješenju problema konstrukcijom

odgovarajuće arhitekture. Dakle, činjenica da su ove mreže konstruisane na dati način utiče na njihovu efikasnost pri rješavanju problema. Jasno je da konstrukcija MTL mreža zahtijeva i određene specifičnosti u načinu obučavanja, pa su, prema tome, ovdje izložene samo arhitekture koje ne zahtijevaju promjenu paradigme pri obučavanju.

Konačno, arhitektura mreže u značajnoj mjeri diktira njenu efikasnost u rješavanju problema koji je dat i potrebno je adekvatno konstruisati arhitekturu da bi bila podesna i za problem koji se rješava i za planirani način obučavanja. Iako je moguće automatski generisati arhitekture mreža [136], a naknadno određivati smisao izbora slojeva i njihovog povezivanja, praktično je najefikasnije konceptualno modelovati problem, pa projektovati arhitekturu na osnovu domenskog znanja problema.

3.6 HARDVERSKA PODRŠKA I ADAPTACIJE ZA HARDVER

Prirodni paralelizam većine individualnih operacija koje se mogu izdvojiti u posebne slojeve sugeriše mogućnost njihovog izvršavanja na masovno paralelnim hardverskim arhitekturama.

Već pojavom prvih arhitektura konvolucionih neuronskih mreža [9], postalo je jasno da se performanse obučavanja i inferencije neuronskih mreža mogu drastično ubrzati korištenjem grafičkih procesora, čiji je hardverski paralelizam izuzetno pogodan za izvršavanje operacija nad tenzorski strukturisanim podacima, kakvi su audio zapisi i slike. Prvobitno, korišteni su grafički šejderski programi, dok se nije pojavio koncept GPGPU (eng. *General Purpose Graphics Processing Unit*) arhitektura i izračunskih šejdera.

Savremeni grafički hardver uključuje i posebnu vrstu tenzorskih jezgara kao i instrukcijski set za rad sa neuronskim mrežama, isključivo konstruisan za paralelno izvršavanje neuronskih mreža u GPGPU kontekstu. Razvoj ovakvog hardvera uveliko je pomogao izuzetno brzoj ekspanziji oblasti mašinskog učenja. Proizvođači ovakvog hardvera obično isporučuju i odgovarajći aplikativni softver i biblioteke koje maksimiziraju iskorištenje specijalizovanog hardvera na njihovim grafičkim uređajima [137] [138]. Većina savremenih alata i biblioteka za rad sa neuronskim mrežama koristi ove biblioteke i pogodnosti hardvera adaptiranog za mašinsko učenje [139].

3.6.1 Paralelizacija i distribucija

Pored potrebe za tačnošću neuronskih mreža, s obzirom na povećanje broja parametara i kompleksnosti njihovih arhitektura, javlja se i potreba za ubrzavanjem njihovog izvršavanja,

i pri obučavanju i pri inferenciji. Očigledan način, s obzirom na tip i strukturu podataka sa kojima neuronske mreže operišu, je paralelizacija na jednoj mašini ili distribucija na klaster. Ovdje se pojavljuju značajni problemi sinhronizacije i komunikacije između individualnih radnih jedinica koji potencijalno mogu da utiču na konvergenciju mreže, budući da je SGD suštinski sekvencijalna operacija.

Kada se radi o paralelizaciji za više procesora, tada je vrlo efikasan algoritam takozvani Hogwild! algoritam. Ovaj algoritam je pogodan kada se radi o rijetko popunjenim podacima, gdje procesori koji pristupaju dijeljenoj memoriji bez zaključavanja neće često prepisati korisne podatke [140]. Hogwild! pristup izvodi korake stohastičkog gradijentskog spuštanja na različitim procesorima istovremeno, ažurirajući parametre sekvencijalno. Iako naizgled naivan, ovaj pristup rijetko generiše slučajeve utrkivanja, jer se oslanja na činjenicu da su ulazni podaci rijetko popunjeni, odnosno da je većina dobijenih delta vrijednosti bliska nuli.

Downpour SGD [61] je varijanta stohastičkog gradijentskog spuštanja razvijena od strane Google kompanije kao prethodnik Tensorflow [124] frejmvorku, za takozvani DistBelief frejmvork. Ovaj frejmvork paralelizuje SGD repliciranjem cijele mreže na više radnih jedinica od kojih je svaka zadužena za ažuriranje određenog dijela parametara. Ovdje, slično kao i kod Hogwild! pristupa, zbog nedostatka sinhronizacije, postoji mogućnost divergencije mreže.

Vrlo efikasan pristup paralelizaciji SGD algoritma su SGD algoritmi tolerantni na kašnjenje koji su adaptacija AdaGrad optimizatora za paralelno okruženje [141].

Tensorflow frejmvork implementira varijaciju na Downpour SGD u svojoj javnoj verziji sa otvorenim kodom, gdje je omogućena distribucija na više GPGPU jedinica, uz implementaciju distribuirane verzije, gdje je implementiran protokol komunikacije porukama između distribuiranih elemenata.

Takođe interesantna implementacija je SGD sa elastičnim prosjekom (eng. *Elastic Averaging SGD*, EASGD) [142] koji koristi elastičnu silu za čuvanje međusobnih relacija između distribuiranih kopija parametara. Ovo dozvoljava kopijama da fluktuišu oko glavne varijable i na taj način bolje istražuju prostor rješenja, potencijalno izlazeći iz lokalnih minimuma.

3.6.2 Hardverska ograničenja

S druge strane, postoje slučajevi kada je neuronske mreže potrebno implementirati na hardverskim uređajima čija je izračunska moć značajno ograničena, kao što su ugrađeni uređaji ili mobilni telefoni. U ovim slučajevima, jedini vid paralelizacije je paralelizacija na jezgre centralnog procesora, što je vrlo često nemoguće. Dodatno, ovi uređaji unose dodatna memorijska ograničenja i na ovaj način ograničavaju konstrukciju neuronskih mreža.

U slučajevima kada je neuronske mreže potrebno isporučiti na ovakve uređaje, obučavanje mreže se gotovo uvijek radi na odvojenim mašinama, osim kada je u pitanju aktivno obučavanje, gdje se generalno preferira obrada na udaljenom serveru.

Ukoliko se mreža isporučuje na uređaj sa ograničenim kapacitetom perzistentne memorije i ograničenim kapacitetom radne memorije, neophodno je sve elemente koji omogućavaju rad mreže svesti na minimalne potrebne. Prvo, neophodno je parametre mreže, dobijene obučavanjem, sačuvati u formatu koji ne opterećuje značajno perzistentnu memoriju. Ovo se može izvesti odsijecanjem irelevantnih vrijednosti pri čuvanju, a potom kompresijom ili čuvanjem samo relevantnih težina. Pored toga, moguće je izvesti i podrezivanje (eng. *pruning*), odnosno algoritamsko uklanjanje veza koje se pokazuju kao beskorisne. S druge strane, smanjivanje veličine datoteke sa parametrima u perzistentnoj memoriji neće nužno umanjiti veličinu modela u radnoj memoriji.

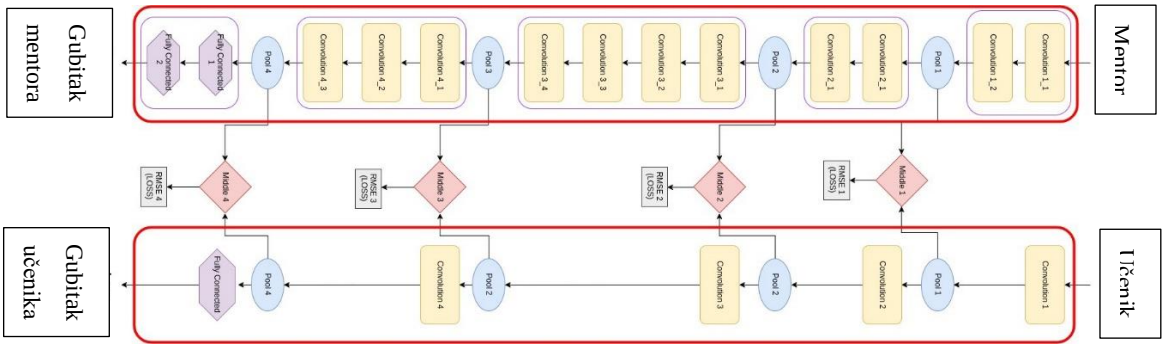
Pored potrebe da se parametri čuvaju u radnoj memoriji, u memoriji mora biti učitan i cijeli frejmvork koji je u stanju da pročita arhitekturu mreže i njene parametre iz datoteke. Obično se koristi minimizirana varijanta frejmvorka korištenog za obučavanje.

Alternativno, cijeli model može da se prevede u mašinski kod i izvršava kao dio isporučenog programskog koda, u kom slučaju nije moguće nakon isporuke ažurirati parametre mreže bez ažuriranja cijele aplikacije. Ovo se, ipak, pokazuje kao najefikasniji način, iako nije dostupan u većini biblioteka za rad sa neuronskim mrežama.

Dodatno, moguće je mreže obučavati alatima različitim od onih koji se koriste za inferenciju, u kom slučaju je neophodno formate datoteka jednog alata pretvarati u format za drugi ili koristiti neki standardizovani format, kao što je ONNX [143] format.

Konačno, potencijalno koristan pristup je i obučavanje manje verzije mreže za rad na mobilnom uređaju, pri čemu se kao posrednik u obučavanju koristi veća mreža slične arhitekture. Ovaj pristup naziva se obučavanje mentorskom mrežom [144] i pokazuje se kao

izuzetno efikasan u obučavanju manjih mreža za zadatak za koji već postoji veća mreža. Slika 3.8 prikazuje obučavanje manje mreže mentorskom mrežom.



Slika 3.8 [145] – *Obučavanje mreže mentorskom mrežom*

3.7 POBOLJŠAVANJE PERFORMANSI MREŽE

3.7.1 Inicijalizacija

Kako bi se osigurala ispravna konvergencija mreže i generalizacija, prije početka obučavanja, vrijednosti parametara mreže obično se inicijalizuju nasumično, sa nekom odabranom distribucijom. Razlog za ovo je narušavanje simetrije, čime se onemogućava specijalizacija više različitih neurona za isti zadatak. Kako su početne vrijednosti parametara različite, promjena njihovih vrijednosti na osnovu gradijenata će biti različita u toku obučavanja.

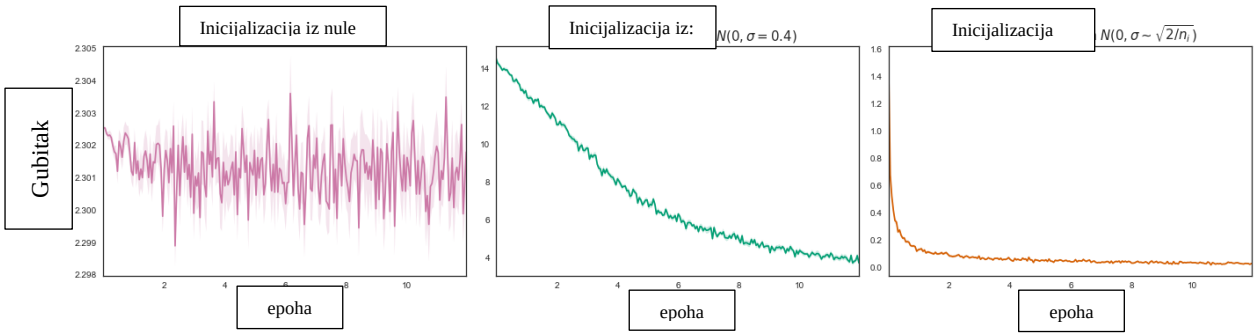
Inicijalizacija parametara na nule daje značajno (često katastrofalno) degradirane rezultate i nerijetko sprečava ispravnu konvergenciju mreže. Različita inicijalizacija je posebno bitna za susjedne neurone, odnosno neurone povezane na isti ulaz, jer ona sprečava stvaranje simetrije i omogućava ispravno ažuriranje težina pri propagaciji unazad [146].

Pri inicijalizaciji parametara mreže, najčešće se koriste normalna i uniformna raspodjela ili druge, njima slične raspodjele, često zavisne od prethodnih slojeva. Najčešće korištene raspodjele su LeCun uniformna raspodjela i Glorot raspodjela (takođe poznata kao Xavier raspodjela) [147].

Pored toga što statistička raspodjela inicijalizovanih vrijednosti parametara zavisi od aktivacione funcije primijenjene na sloju, Xavier raspodjela podrazumijeva da je raspodjela vrijednosti zavisna ne samo od broja ulaza u sloj, već i od broja izlaza [148]. Jednačina 3.3 opisuje skaliranje kod Xavier raspodjele, gdje je w težina koja se ažurira, U generatorska funkcija uniformne raspodjele, a n_j broj neurona u j-tom sloju.

$$w \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right] \tag{3.3}$$

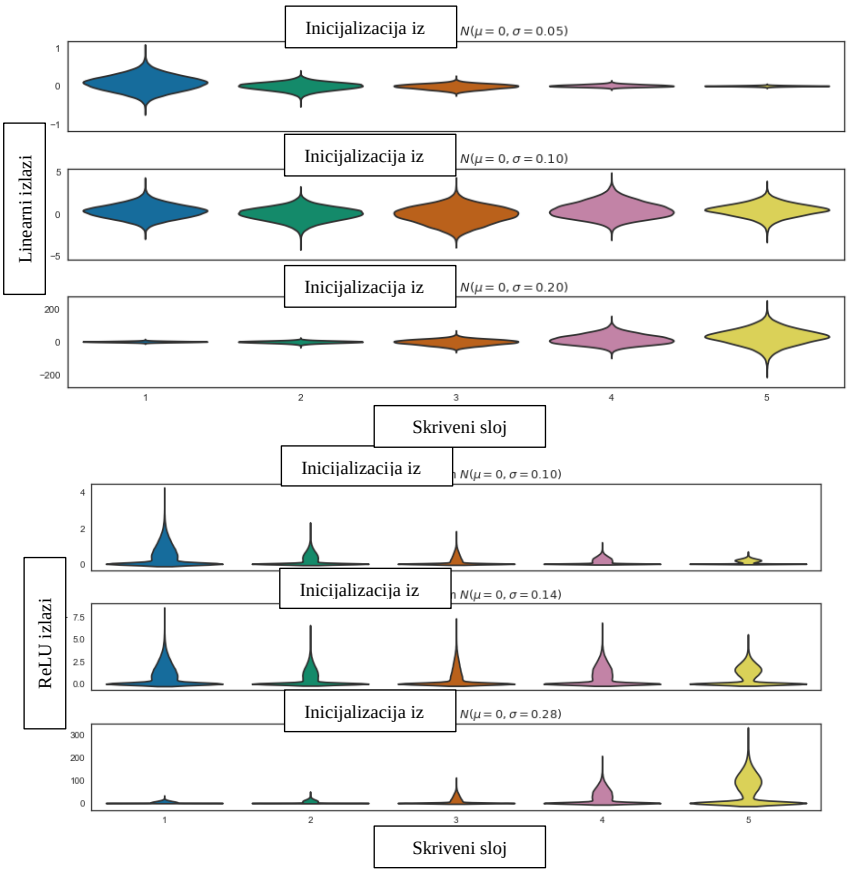
Pored Xavier inicijalizacije, a kao varijacija na nju, koristi se i He inicijalizacija [12]. Ova inicijalizacija je suštinski ista, s tom razlikom što množi cijeli interval konstantnim faktorom 2. Iako je izmjena relativno jednostavna, pokazuje se kao vrlo efikasna u određenim primjenama i često daje bolje rezultate od inicijalizacije Xavier raspodjelom. Slika 3.9 prikazuje razliku ponašanja funkcije gubitka u odnosu na korišteni način inicijalizacije.



Slika 3.9 [149] – *Inicijalizacija na nulu, inicijalizacija normalnom raspodjelom, inicijalizacija normalnom raspodjelom zavisnom od broja ulaza (slijeva udesno)*

Suštinski, idealna inicijalizacija će omogućiti čuvanje statističkih osobina raspodjele vrijednosti obučavanih parametara mreže na različitim slojevima, bez obzira na broj iteracija. Pokazuje se da izbor koeficijenata raspodjele značajno utiče na raspodjelu parametara za vrijeme obučavanja, pri čemu ne postoje idealne vrijednosti koeficijenata, već ih je neophodno podešavati na osnovu arhitekture mreže, kako sugerišu autori pomenutih inicijalizacionih šema.

Slika 3.10 prikazuje statističku raspodjelu aktivacija (intenzitet aktivacije u odnosu na broj neurona) na pet skrivenih slojeva jednostavne FFNN mreže, sa različitim oblicima inicijalizacije. Na istoj slici se vidi i ponašanje izlaza neurona kada se koristi ista inicijalizaciona šema, ali ReLU aktivaciona funkcija. Sa slike se vidi da odabir parametara raspodjele za inicijalizaciju značajno utiče na konzistentnost raspodjele izlaza za različite slojeve.



Slika 3.10 [149] – *Inicijalizacija sa Gausovom raspodjelom i σ=0.05, σ=0.1 i σ=0.2 za linearnu aktivacionu funkciju (prva tri reda) i σ=0.1, σ=0.14 i σ=0.28 za ReLU aktivacionu funkciju (posljednja tri reda) na svakom od pet skrivenih slojeva mreže*

3.7.2 Podešavanje hiperparametara

Hiperparametrom se naziva svaki parametar mreže koji se podešava za vrijeme obučavanja, a koji se ne koristi pri inferenciji. Najčešći hiperparametri su parametri optimizatora, odnosno koeficijenti optimizatorskih jednačina, koeficijent učenja i raspored promjene koeficijenta učenja u odnosu na broj iteracije ili epohe, veličina grupe kao i parametri slojeva mreže (kao što su, na primjer, kod konvolucionih slojeva, korak, broj mapa obilježja ili parametar dilacije). Međusobno povezivanje (rutiranje) slojeva se odnosi na arhitekturu mreže i rijetko se podvodi pod termin hiperparametar.

Ovdje vrijedi napomenuti da se pod termin *hiperparametar* vrlo često podvode i apstraktni elementi arhitekture neuronske mreže, kao što su broj slojeva, broj mapa obilježja u različitim slojevima, tipovi slojeva, kao i sam proces pronalaženja odgovarajuće mreže, odnosno proces promjene rutiranja u mreži i definisanja slojeva. Kako bi se izbjegli problemi sa postojećom definicijom ovog termina u literaturi, za svrhe ove disertacije termin *hiperparametar* je korišten u užem kontekstu definisanom iznad.

Na konvergenciju pri obučavanju, pored odgovarajućeg odabira arhitekture i parametara slojeva, najviše utiče izbor koeficijenta učenja i parametara optimizatora. Često se hiperparametri, osim ako nisu na neki način povezani (na primjer, optimizatorskom jednačinom), mogu podešavati nezavisno, što značajno smanjuje prostor pretrage po hiperparametrima za vrijeme razvoja i validacije arhitekture mreže, jer se optimalni hiperparametri mogu pronalaziti pojedinačno. Kada se optimizuje jedan hiperparametar, obično se može nastaviti sa optimizacijom ostalih bez potrebe za ponovnim podešavanjem prethodnih na osnovu ostalih.

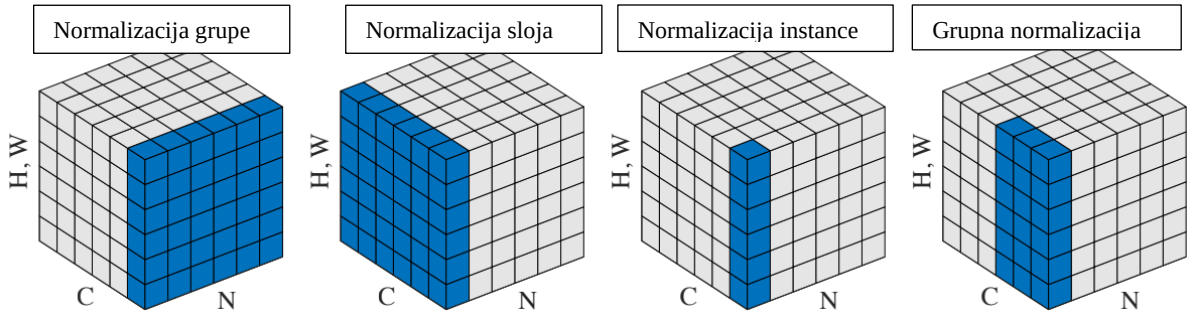
Ukoliko optimizovanje svih hiperparametara nije praktično izvedivo, optimizacija koeficijenta učenja najčešće daje najvidljivije rezultate. U praksi se često pokazuje da je najefikasnije koeficijent učenja postepeno smanjivati sa brojem epoha, gdje se kao najefikasnije pokazuju stepenasto spuštanje i eksponencijalno spuštanje.

Uopšteno, odabir hiperparametara se najčešće radi serijom eksperimenata sa ručnim varijacijama vrijednosti, na osnovu opservacije performansi mreže. Kako bi se osiguralo da proces izmjena arhitekture mreže ne dovodi do specijalizacije za skup za obučavanje, korekcija performansi mreže treba da bude izvedena nad validacionim skupom, a konačna evaluacija performansi nad testnim skupom. S druge strane, proces validacije je moguće automatizovati ili sekvencijalnom pretragom i automatizovanim obučavanjem ili nekom optimizovanom pretragom, kakav je, na primjer, genetički algoritam [150].

3.7.3 Normalizacija

Ranije je pomenuto da normalizacija ulaznih podataka može da pomogne konvergenciji mreže i ubrza obučavanje. Normalizacija elemenata tenzora može da se koristi i nad izlaznim i nad ulaznim tenzorima pojedinačnih slojeva.

Tipično se razlikuju četiri vrste normalizacije: normalizacija grupe (eng. *batch normalization*), normalizacija sloja (eng. *layer normalization*), normalizacija instance (eng. *instance normalization*) [151] i grupna normalizacija (eng. *group normalization*). Slika 3.11 prikazuje elemente koji se normalizuju u tenzoru, za različite tipove normalizacije.



Slika 3.11 [152] – *Normalizacija grupe, normalizacija sloja, normalizacija instance i grupna normalizacija (slijeva udesno; C predstavlja broj kanala ili mapa obilježja kod konvolucije, N broj uzoraka u grupi, a H,W elemente uzorka objedinjene na jednu osu radi trodimenzionog prikaza tenzora koji je tipično četvorodimenzion).*

Normalizacija sloja vrši normalizaciju po svim elementima jednog datog uzorka (u slučaju trodimenzione konvolucije, normalizacija se radi nad elementima po osama broja kanala, širine i dužine). Grupna normalizacija vrši normalizaciju zadatog podskupa jednog datog uzorka, particionisanog po osi kanala odnosno mapa obilježja. Grupna normalizacija sa particionisanjem na jedan skup postaje normalizacija sloja.

Normalizacija instance vrši normalizaciju po osama nezavisno od mapa obilježja, odnosno samo po osama dimenzija ulaznog podatka.

Normalizacija grupe skalira sve delte parametara u jednoj iteraciji (odnosno, u trenutnoj grupi), na svakom pojedinačnom sloju, izračunatom normom svih parametara u trenutnoj grupi. Normalizacija se obično vrši prije slanja izlaza neurona aktivacionim funkcijama. Dodatno, kako na ovaj način izlaz svakog neurona zavisi od izlaza svih ostalih iz sloja, za datu grupu (odnosno, dijeli se sa normom svih izlaza za dati sloj, za sve primjere iz date grupe), normalizacija grupe ima regularizacioni efekat.

Budući da se između različitih parova ulaz-izlaz iz skupa za obučavanje vrijednosti izlaza individualnih neurona mogu značajno mijenjati, normalizacija grupe, koja se izvodi nad svim primjerima u grupi, smanjuje ove varijacije između različitih ulaza i vrši regularizaciju i na nivou iteracija, ograničenu brojem primjera u grupi.

Ukoliko je korištena normalizacija grupe pri obučavanju, potrebno je ovo uzeti u obzir pri inferenciji, jer ovako obučavana mreža zahtijeva inferenciju u grupama, a ne individualno. Kako se normalizacija radi nad grupama izračunavanjem srednje vrijednosti i varijanse grupe, ovo se može kompenzirati izračunavanjem srednje vrijednosti i varijanse između svih grupa, i korištenjem dobijene vrijednosti za sve buduće inferencije, kao član za normalizaciju. Ovako

je pri inferenciji veličina grupe svedena na jedan element, ali su iskorišteni normalizacioni faktori dobijeni računanjem statistike nad cijelim skupom za obučavanje [71].

4 OBLASTI PRIMJENE I SPECIJALIZACIJA

4.1 PREGLED

U prethodnim glavama data je neophodna osnova za analizu najkorištenijih arhitektura neuronskih mreža i njihove funkcionalnosti. U ovoj glavi biće predstavljeni i detaljno analizirani mehanizmi rada najčešće korištenih i istorijski najznačajnijih arhitektura neuronskih mreža sa ciljem pronalaženja zajedničkih pravila koja su inžinjeri i istraživači koji su razvili ove arhitekture koristili. Koncepti i ideje korištene i izvedene pri razvoju, obučavanju i upotrebi ovih modela iskorištene su za konstrukciju metodologije razvoja arhitekture mreža predstavljene u Glavi 5 i iskorištene za implementaciju rješenja predstavljenog u Glavi 6.

Poglavlje 4.2 predstavlja klasifikacione modele, daleko najistraženije i najkorištenije u praksi, te analizira njihovu funkcionalnost i ideje iskorištene pri njihovom projektovanju, uključujući modularizaciju, te se dotiče ideje učenja sa transferom, koje je detaljno analizirano u Glavi 5. Klasifikacione arhitekture su od najvećeg interesa za ovu disertaciju, jer implementirano rješenje izvodi suštinski taj zadatak.

U poglavljima 4.3 i 4.4, na sličan način, predstavljeni su modeli za detekciju objekata i segmentaciju slike, respektivno. Ova dva poglavlja usko su vezana za materijal u poglavlju 4.2 i takođe su izuzetno značajna za rješenje implementirano u okviru ove disertacije.

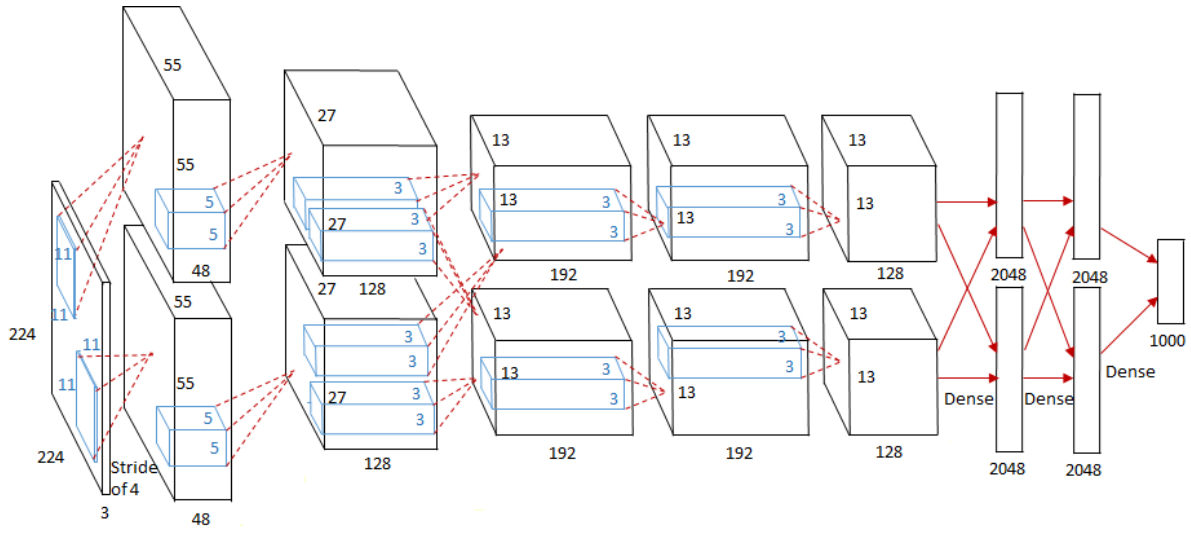
Poglavlje 4.5 predstavlja arhitekture za generisanje sadržaja. Ove arhitekture nisu suštinski vezane za dato rješenje, ali su neophodne za potpunost metodologije na osnovu koje je razvijeno rješenje, te su ovdje date radi nekoliko koncepata koji mogu biti predstavljeni samo kod ovih arhitektura, uključujući augmentaciju korištenjem generatorskih mreža za proširivanje ili zamjenu skupova za obučavanje.

4.2 KLASIFIKACIJA

Daleko najistraženiji i najkorišteniji modeli neuronskih mreža su klasifikacioni modeli koji ulaze različitih tipova preslikavaju u labele koje označavaju kategorije ili klase. Naravno, ovo ne znači da je klasifikacija najčešće implementiran zadatak, već da ovi modeli, koji su, istorijski gledano, prvi razvijeni kako je oblast mašinskog učenja ponovo počinjala da raste, imaju vrlo široku upotrebu i u oblastima koje ne uključuju nužno klasifikaciju, zbog njihove podobnosti za adaptaciju u ovim oblastima.

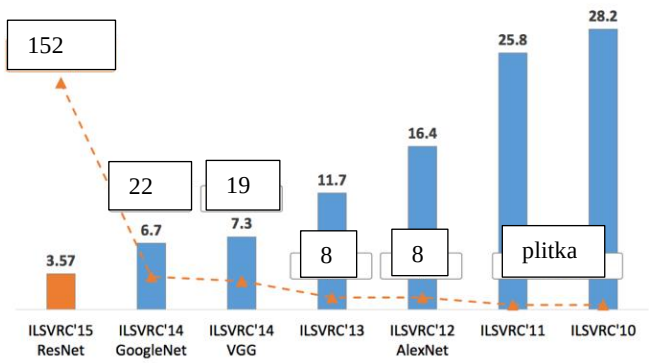
Iz istorijskih razloga, bitno je napomenuti postojanje **LeNet-5** [9] mreže, sedmoslojne konvolucione neuronske mreže koja je predstavljala osnov za razvoj novih arhitektura konvolucionih neuronskih mreža. Ova mreža, predstavljena Slikom 2.3, je suštinski jednostavna konvoluciona neuronska mreža za klasifikaciju rukom pisanih cifara koja operiše nad ulaznim slikama dimenzija 32×32.

Kao nadogradnja na LeNet, vrlo efikasan model vještačke neuronske mreže za klasifikaciju slika je **AlexNet** [153]. Ova mreža predstavlja varijaciju na LeNet, sa značajnim brojem manjih modifikacija koje su zajedno doprinijele povećanju top-5 tačnosti sa 74% na 84.7% greške na ImageNet skupu. Ova mreža prihvata ulaznu sliku dimenzija 224×224 i vrši propagaciju unaprijed kroz seriju slojeva prikazanih Slikom 4.1. Ova mreža koristila je konvolucione slojeve 11×11, 5×5 i 3×3, agregacija po maksimumu, odbacivanje i ReLU aktivacione funkcije. Pored toga, ova arhitektura sastoji se od dva stabla slojeva, jer je obučavana paralelno na dva grafička procesora.



Slika 4.1 [154] – *Arhitektura AlexNet mreže*

Po savremenim standardima, AlexNet mreža je relativno malena, ali se pokazuje kao pogodna za jednostavnije zadatke za klasifikaciju, pogotovo kada se specijalizuje za zadatak korištenjem modela predobučavanog na ImageNet skupu. Slika 4.2 prikazuje odnos broja slojeva, tačnosti klasifikacije i godine kada je odgovarajući model razvijen, gdje je godina sadržana u nazivu ILSVRC [155] takmičenja.



Slika 4.2 [154] – *Odnos broja slojeva, tačnosti klasifikacije i godine razvoja modela neuronske mreže*

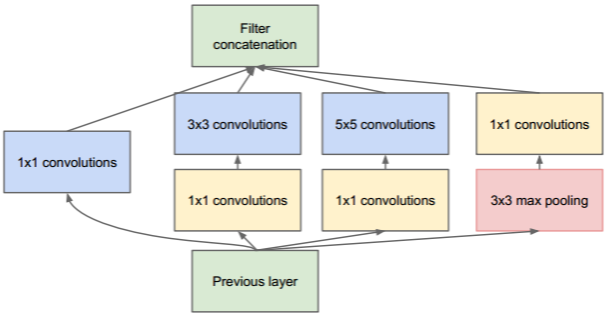
Za razliku od razgranate AlexNet mreže, **VGGNet** mreža [156], data Slikom 4.3, je konvoluciona neuronska mreža bez grananja za 16 slojeva, nekad zvana i VGG16 (kada se koristi konfiguracija D, sa Slike 4.3). Ova mreža u potpunosti izbacuje velike konvolucione kernele, koristeći samo kernele dimenzija 3 × 3. Pored ove izmjene VGG mreža povećava broj filtara po sloju. VGG mreža je prezentovana sa nekoliko konfiguracija, kao na Slici 4.3, gdje su najkorištenije konfiguracije D i E. Pored ovoga, VGG uvodi koncept modula koji je u slijedećim arhitekturama vrlo često korišten. Moduli u arhitekturama neuronskih mreža predstavljaju podsegmente mreža koji se ponavljaju ili ponovo koriste. Ova arhitektura je najčešće korištena pri izvlačenju obilježja sa slika [154].

Konfiguracija konvolucione mreže					
A	A-LRN	B	C	D	E
11 slojeva	11 slojeva	13 slojeva	13 slojeva	16 slojeva	19 slojeva
Ulaz (224 × 224 slika)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Slika 4.3 [157] – *Arhitektura VGG mreže*

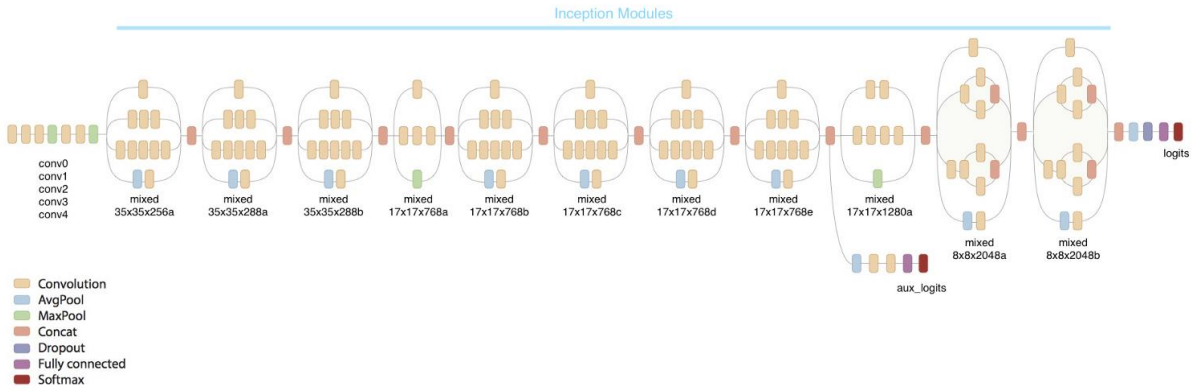
Pri obučavanju VGG mreže, pokazalo se efikasnije obučavati manje konfiguracije mreža sa manje parametara, a potom njihove težine koristiti u ponovnom obučavanju većih mreža (koje, uz specifične dodatne slojeve, sadrže iste slojeve kao i manje varijante mreže). Ovaj vid obučavanja naziva se obučavanje sa predobučavanjem i relativno rijetko se koristi u ovom obliku (prenošenje parametara sa različitih varijanti iste mreže, na istom zadatku), jer se pokazuje da su Xavier i He inicijalizacije bolje ili ekvivalentne predobučavanju [158]. Loša strana upotrebe ove arhitekture je u tome što njeno obučavanje može da traje izuzetno dugo, te zahtijeva i do 574MB memorije (u slučaju VGG19 arhitekture) za čuvanje parametara, što je čini relativno neupotrebljivom u primjenama sa ograničenim hardverom.

Na istom ImageNet [155] takmičenju na kom je VGGNet pokazala izuzetnu tačnost i osvojila drugo mjesto, **GoogleNet** ili Inception-v1 [159] mreža osvojila je prvo mjesto sa top-5 greškom od 6.67%. Kako je već nagoviješteno u poglavlju 3.4, ova arhitektura uvela je nekoliko fundamentalnih izmjena u konstrukciji mreže koje su omogućile povećanje tačnosti, a smanjenje broja parametara. Pored pomenute globalne prosječne agregacije, za ovu mrežu je konstruisan poseban modul, predstavljen Slikom 4.4. Ovaj modul aproksimira rijetko popunjenu CNN mrežu korištenjem kombinacije malih punih CNN mreža (gdje je rijetko popunjena mreža ona sa većim brojem nultih parametara, dok je puna mreža ona kod koje su svi parametri upotrebljivi). Kako je ranije rečeno, prepoznato je da se jedan linearni filtar može zamijeniti multilinearnim perceptronom, koji bi trebalo da predstavlja rijetko popunjenu matricu. Kako u to vrijeme nije postojala odgovarajuća hardverska podrška za efikasno izračunavanje proizvoda sa rijetko popunjenom matricom, za Inception mrežu konstruisan je aproksimativni ekvivalentan modul, koji se pokazao kao brži i dovoljno efikasan.



Slika 4.4 [157] – *Arhitektura Inception modula GoogleNet mreže*

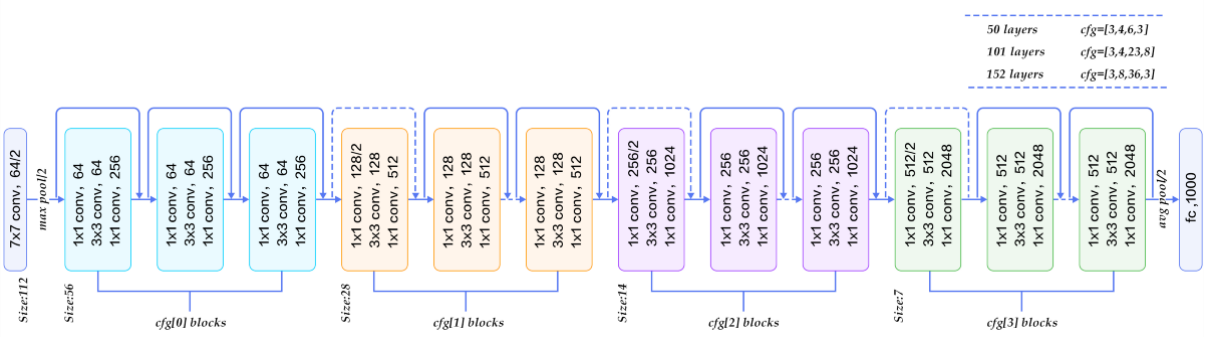
Suštinski, GoogleNet mreža sastoji se od serije ovakvih Inception modula. Konkretno, GoogleNet mreža sastoji se od ulaznog sloja, pet konvolucionih slojeva, serije Inception modula i sloja za globalnu prosječnu agregaciju, kako je dato Slikom 4.5 [160].



Slika 4.5 [160] *– Arhitektura Inception-v1 mreže*

Iako na prvi pogled ova mreža izgleda jednostavno, svaki modul je u stanju da uči obilježja različitih veličina, koduje invarijantnost na skaliranje i smanjuje izračunske zahtjeve korištenjem konvolucija kernelom 1×1.

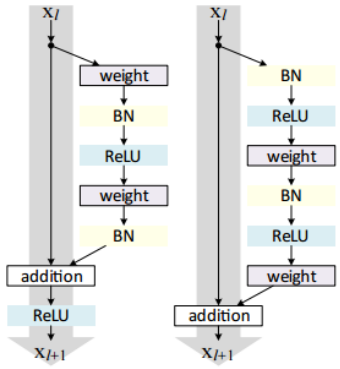
ResNet arhitektura [104], odnosno rezidualna neuronska mreža, pored ideje mreže u mreži iskorištene kod Inception mreže, koristi preskačuće veze da propagira informacije sa ulaza na dublje module. Ova arhitektura se pokazala kao efikasna u prevazilaženju ljudske tačnosti na ImageNet skupu i prikazana je Slikom 4.6.



Slika 4.6 – Arhitektura ResNet mreže

Ova arhitektura sadrži 152 sloja, a njeno obučavanje je omogućeno zahvaljujući preskačućim vezama za čije je uvođenje inspiracija bila klasična arhitektura rekurentnih neuronskih mreža. Ranije je pomenuto da se RNN mreže mogu „razmotavati“ i koristiti sekvencijalno. Taj isti koncept razmotavanja rekurentnih mreža iskorišten je u ResNet arhitekturi kako bi se vrijednosti sa ulaza propagirale dublje u mrežu i kako bi se gradijenti sa

izlaza lakše propagirali unazad, bez nestajućeg gradijenta. Eliminacija problema nestajućeg gradijenta i uvođenje He inicijalizacije omogućilo je konstrukciju ovako duboke arhitekture, pri čemu je ponovo demonstrirana činjenica da dubina mreže najznačajnije utiče na tačnost. Mreža je kasnije nadograđena jediničnim mapiranjima na nivou modula [161]. Slika 4.7 predstavlja rezidualni modul originalne mreže i nadograđeni rezidualni modul.



Slika 4.7 – Originalni rezidualni modul (lijevo) i nadograđeni rezidualni modul (desno)

Iako, na prvi pogled, ResNet mreža djeluje kompleksnija od GoogleNet mreže i VGG mreže, njen broj parametara je manji nego kod ove dvije mreže, jer ne koristi potpuno povezane slojeve i ne koristi veliki broj filtara po individualnom sloju, što je čini upotrebljivom na uređajima sa ograničenim hardverom.

GoogleNet Inception mreža je razvijana i u narednim godinama, do svoje četvrte verzije, Inception-v4 i kombinacije sa ResNet arhitekturom u Inception-ResNet arhitekturu. Ove arhitekture nastavljaju sa sličnom idejom iz originalne Inception arhitekture i ResNet arhitekture da bi se postigla top-5 tačnost od 3.08% na ImageNet skupu [162].

Za primjene sa ograničenom memorijom, praktično su upotrebljive i Inception-v3 [163] mreža i Xception [164] mreža koje su varijacije na prethodno date arhitekture, sa parametrima mreže koji zauzimaju ispod 100MB. Za ovu primjenu, ukoliko je memorija značajno ograničena, upotrebljiva je i SqueezeNet [165] mreža čiji parametri zauzimaju ispod 5MB memorijskog prostora. Ova mreža, iako može da postiže rezultate uporedive sa AlexNet mrežom, često je vrlo nepraktična za obučavanje, jer je neophodno obučavati ili u mentorskom režimu ili u režimu specijalizovanom za dati problem.

4.3 DETEKCIJA

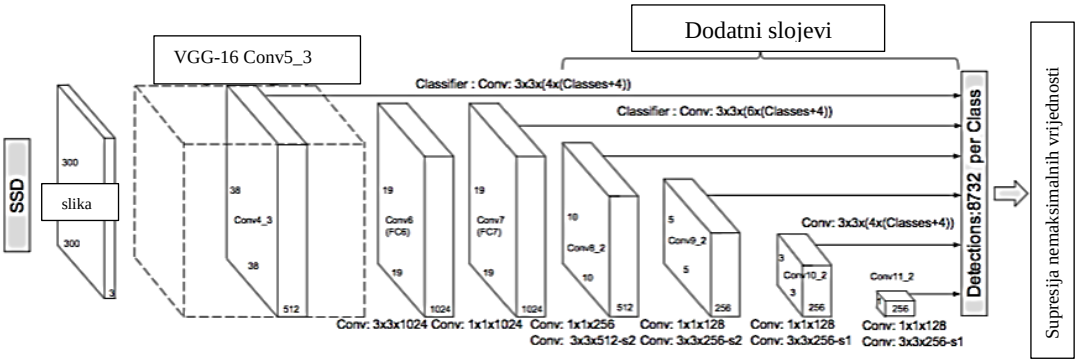
Budući da je već ustanovljeno da su duboke neuronske mreže bolje u specifičnim zadacima klasifikacije od ljudi, prirodno je postaviti pitanje o tome gdje se ovi modeli dodatno mogu koristiti. Iako su pomenuti modeli u stanju da klasifikuju ulazne slike, ljudska tačnost u klasifikaciji je samo posljedica kompleksnijeg mehanizma koji uključuje i segmentaciju vizuelnog polja i klasifikaciju individualnih segmenata. Ovo znači da su ljudi u stanju da detektuju instance individualnih klasa na slikama, te da odrede njihovu lokaciju u sopstvenom vizuelnom polju.

Kako već postoje vrlo efikasni modeli za klasifikaciju individualnih slika, prirodan sljedeći korak je razvijanje modela koji su u stanju da detektuju instance klasa na ulaznim slikama. Postoji nekoliko glavnih modela koji se u praksi najčešće koriste za ovu svrhu.

Regionalna konvoluciona neuronska mreža ili **R-CNN** [166] je jedan takav model, zajedno sa njemu sličnim modelima: Brzi R-CNN (eng. *Fast R-CNN*) [111] i Brži R-CNN (eng. *Faster R-CNN*) [167], konstruisan s ciljem ostvarivanja detekcije objekata u realnom vremenu. Budući da je korištenje klasične konvolucione neuronske mreže nepraktično zbog cijene izračunavanja i činjenice da izlaz može biti promjenljive dužine (jer je cilj određivanje postojanja i položaja objekta, obično obuhvatajućim pravougaonikom), ova mreža i njene varijante uvode novi algoritam selekcije odgovarajućih regiona. Kao korak predobrade, algoritamski se pronalazi 2000 kandidatskih regiona na ulaznoj slici, nakon čega se svaki od regiona individualno klasifikuje neuronskom mrežom i SVM klasifikatorom. Ovdje se CNN mreža koristi kako bi se izvukli 4096-dimenzioni vektori obilježja koji se potom klasifikuju SVM klasifikatorom (gdje klase određuju prisustvo ili odsustvo objekta).

Glavni problemi kod ovog pristupa su u vremenu potrebnom da se svi regioni klasifikuju i u činjenici da obučavanje može da traje izuzetno dugo. Ovaj model se pokazao kao neprimjenjiv u realnom vremenu. Pored toga, algoritam za predselekciju regiona je klasičan, deterministički, algoritam, bez učenih parametara. Implementacija Brzog R-CNN izbacuje ovaj algoritam i koristi konvolucionu neuronsku mrežu za određivanje mapiranja obilježja sa ulazne slike, na osnovu kojih se određuju regioni od interesa koji se onda skaliraju u kvadratni oblik i šalju ostatku mreže. Ovdje je, iako su poboljšane performanse, usko grlo i dalje predstavljao algoritam za selekciju regiona. Kod Bržeg R-CNN, ovaj problem je prevaziđen uvođenjem dodatne mreže koja služi za selekciju regiona. Ovo je jedini od tri pomenuta pristupa koji je upotrebljiv u detekciji u realnom vremenu [168].

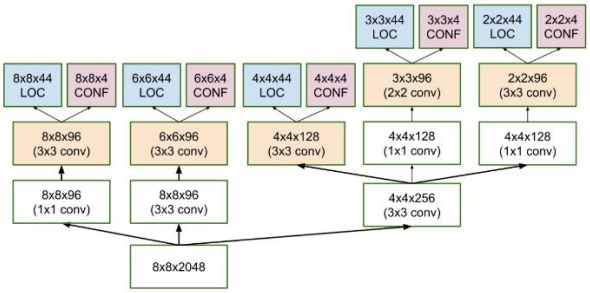
U praksi, ove mreže se rijetko koriste jer su razvijene arhitekture koje su daleko efikasnije u rješavanju istog zadatka. Jedna takva arhitektura je višeregionski detektor jednim slikanjem (eng. *Single Shot Multibox Detector*, **SSD**) [169]. Ovaj detektor inspirisan je R-CNN mrežom i VGG arhitekturom [170] [171]. Karakterišu ga dvije osobine: detekcija u jednom koraku inferencije i višeregionski algoritam za regresiju obuhvatajućih pravougaonika. Slika 4.8 prikazuje arhitekturu SSD mreže.



Slika 4.8 – Arhitektura SSD mreže

Pored korištenja VGG16 mreže, SSD mreža mijenja potpuno povezane slojeve sa šest pomoćnih konvolucionih slojeva koji služe za izdvajanje obilježja različitih veličina i slanje tih obilježja preskačućim vezama na konačni detektor. Tehnika detekcije obuhvatajućih pravougaonika inspirisana je MultiBox [172] detekcijom, koja se zasniva na konvolucionoj neuronskoj mreži koja koristi modul sličan Inception modelu.

Slika 4.9 prikazuje MultiBox modul za detekciju i lokalizaciju. Pri obučavanju koriste se dvije funkcije gubitka: prva za pouzdanost u tačnost izračunatog obuhvatajućeg pravougaonika, implementirana kao među-entropijska funkcija gubitka i druga za mjerenje daljine lokacije predviđenog obuhvatajućeg pravougaonika od onog u labeliranom skupu, gdje je korištena L2 norma.



Slika 4.9 – MultiBox modul za detekciju i lokalizaciju; slojevi označeni sa LOC i CONF predstavljaju one koji se optimizuju za lokacionu krešku i pouzdanost u tačnost, respektivno

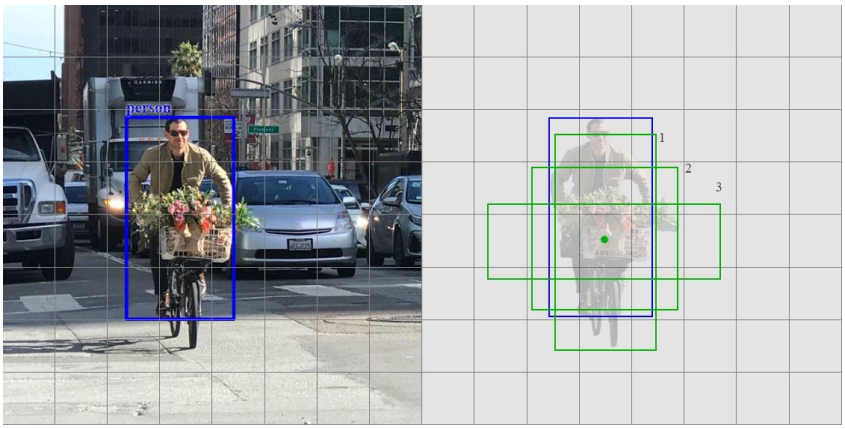
Objedinjena funkcija gubitka formulisana je kao kombinacija ove dvije funkcije gubitka, pri čemu se gubitak po lokaciji skalira parametrom γ, kako je dato jednačinom (4.1).

$$E(X, \theta) = E_1(X, \theta) + \gamma E_2(X, \theta) \tag{4.1}$$

Ova funkcija gubitka primjenjiva je za MultiBox detektor, koji ne izvodi klasifikaciju, dok SSD mreža mora da izvodi klasifikaciju, pa je kod SSD implementacije funkcija gubitka za lokaciju zamijenjena L1 normom, te je za svaki obuhvatajući pravougaonik dodat i vektor koji određuje pripadnost klasi, kao što je slučaj u klasičnim klasifikacionim modelima.

Kako bi se smanjio prostor pretrage za individualne obuhvatajuće pravougaonike, njihov oblik i pozicija izračunavaju se ne u globalnom koordinatnom sistemu slike, već u koordinatnom sistemu koji zavisi od najvjerovatnijeg oblika (odnosno, očekivanog oblika) obuhvatajućeg pravougaonika za dobijenu klasu. Ovaj najvjerovatniji oblik određuje se ručno i definiše u fazi obučavanja. Razlog za ovo je činjenica da u realnosti različite klase objekata imaju specifičan odnos visine i širine.

Za vrijeme obučavanja, predikcije se klasifikuju samo kao pozitivne ili negativne, a samo se pozitivne predikcije uključuju u izračunavanje funkcije gubitka za lokaciju, pri čemu se predikcija smatra pozitivnom ukoliko je ručno određeni podrazumijevani obuhvatajući pravougaonik (a ne predviđeni pravougaonik) poklopljen sa labeliranim, sa odnosom površine njihovog presjeka i unije većim od 0.5. Slika 4.10 prikazuje ovo poređenje.

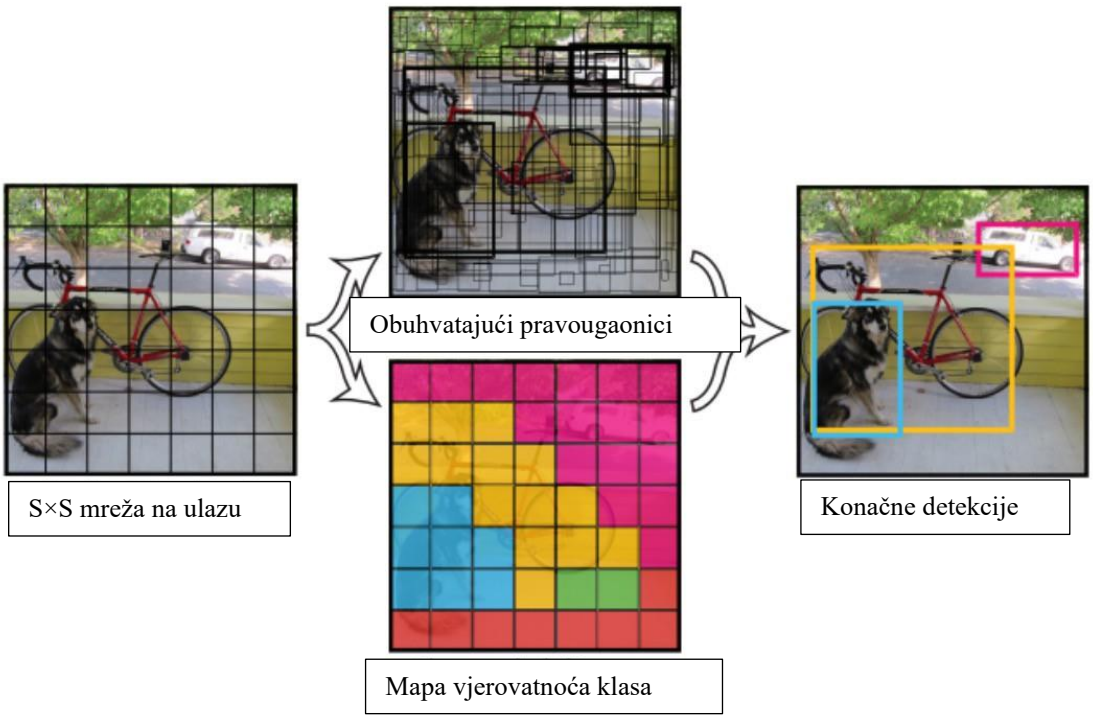


Slika 4.10 – Labelirani primjer (lijevo) i podrazumijevani obuhvatajući pravougaonici za svaku od tri klase koje se detektuju (desno)

Mape obilježja više rezolucije u SSD arhitekturi zadužene su za detekciju manjih objekata, dok su mape obilježja niže rezolucije zadužene za detekciju većih objekata.

Pored ovog pristupa, vršena je augmentacija nad cijelim skupom za obučavanje, korištenjem isijecanja podslika (eng. *cropping*) iz slika iz skupa za obučavanje, korištenjem nasumičnog koeficijenta omjera. Pored toga, ovaj pristup koristi još nekoliko tehnika pri obučavanju, uključujući traženje negativnih primjera koje je teško klasifikovati (eng. *hard negative mining*) i supresiju nemaksimalnih vrijednosti (eng. *Non-Maximum Suppression*, NMS).

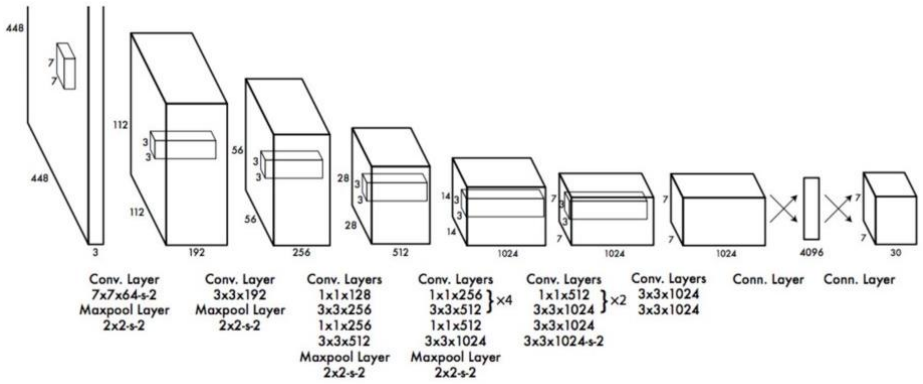
Kao nadogradnja na SSD arhitekturu, formirane su **YOLO** [173] i YOLO9000 [174] arhitekture koje pokazuju dovoljno dobre performanse za izvršavanje u realnom vremenu i brže, odnosno 45 frejmova u sekundi, na testiranom hardveru [175]. Ova arhitektura, slično kao i SSD, modeluje problem detekcije kao regresioni problem generišući predikcije unutar definisane mreže nad ulazom i potom spajajući vrijednosti na kraju, kako je ilustrovano Slikom 4.11.



Slika 4.11 – Regresioni model detekcije kod YOLO arhitekture

Arhitektura YOLO mreže data je Slikom 4.12. Ova arhitektura generiše više predikcija obuhvatajućih pravougaonika po ćeliji mreže, gdje se za vrijeme obučavanja svaki prediktor specijalizuje za samo jednu vrstu objekta, najvjerovatniju za datu ćeliju. Ovo, prema autorima, dovodi do specijalizacije između prediktora. S druge strane, ovo unosi značajna prostorna

ograničenja i čini mrežu neefikasnom u prepoznavanju većeg broja objekata na bliskim lokacijama.



Slika 4.12 – Arhitektura YOLO mreže

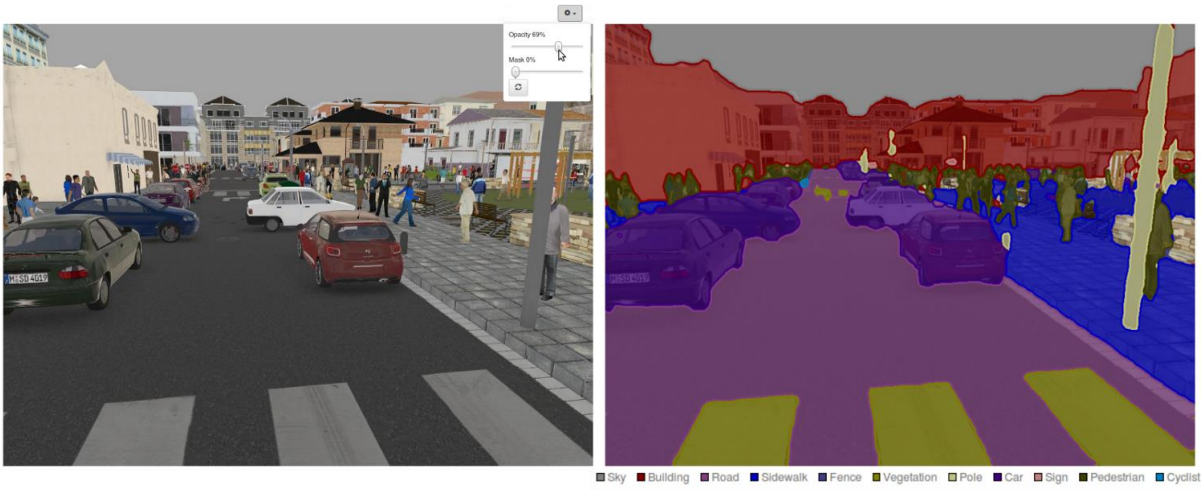
S druge strane, YOLO9000 mreža koristi neke od ideja R-CNN i SSD mreža, uklanjajući potpuno povezane slojeve i koristeći relativni koordinatni sistem za predikcije obuhvatajućih pravougaonika. Pri obučavanju ove mreže, koriste se i drugi pristupi, uključujući normalizaciju grupa, fino podešavanje, i korištenje različitih slika iz skupova za detekciju i klasifikaciju.

Zavisno od konkretne oblasti, obično se koriste SSD i YOLO arhitekture ukoliko je cilj detekcija objekata u realnom vremenu. Mreže sa YOLO arhitekturom se preferiraju ukoliko su performanse od značaja, dok su SSD mreže pogodnije ukoliko je potrebno obučavanje raditi na značajno drugačijim skupovima od onih na kojima će biti testirana, jer su bazirane na VGG mreži, koja tipično dobro generalizuje. Dodatno, SSD pristup je povoljan, jer je moguće promijeniti arhitekturu mreže koja se koristi, kako se može vidjeti sa Slike 4.8. Nerijetko se umjesto VGG mreže u SSD mrežama koristi Inception mreža, ResNet ili neka druga mreža. Ovo zavisi od konkretne primjene i od dostupnosti već obučenih modela klasifikacionih mreža nad problemom slične vrste. Generalno, preferira se SSD sa nekom dubljom mrežom, osim ako performanse nisu od suštinskog značaja.

4.4 SEGMENTACIJA

Nakon detekcije, sljedeći korak, koji približava neuronske mreže ljudskoj perceptivnoj sposobnosti, je segmentacija dobijene slike na regione na kojima se objekti nalaze. Ovaj pristup ne uključuje obuhvatajuće pravougaonike, već zahtijeva određivanje maske za svaku od klasa koja se detektuje i izuzetno je koristan u primjenama kod robotskog vida i automatskog upravljanja, gdje je neophodno imati jasno semantičko razumijevanje svih piksela ulazne slike.

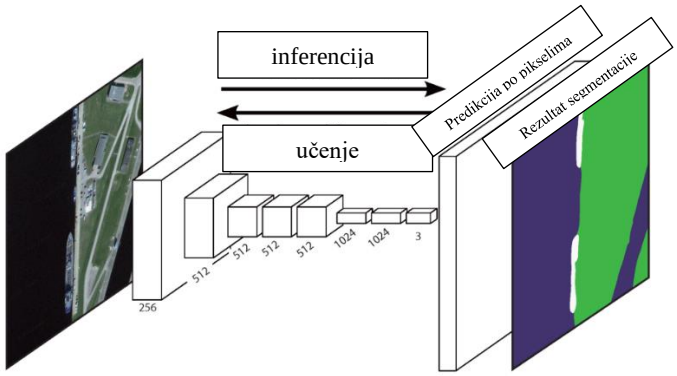
Ovaj pristup naziva se semantička segmentacija. Primjer rezultata semantičke segmentacije dat je Slikom 4.13.



Slika 4.13 [176] – *Semantička segmentacija za autonomna vozila (lijevo: ulazna slika, desno: segmentirana slika)*

Iako se prethodno pomenuta detekcija (primjeri iz poglavlja 4.3) može nazvati i regionskom semantičkom segmentacijom, ona ne segmentiše sliku sa stvarnim obuhvatajućim krivim, i tako nije formalno segmentaciona tehnika. S druge strane, najčešće korišten pristup za stvarnu semantičku segmentaciju su potpuno konvolucione neuronske mreže [177], predstavljene u poglavlju 2.7.

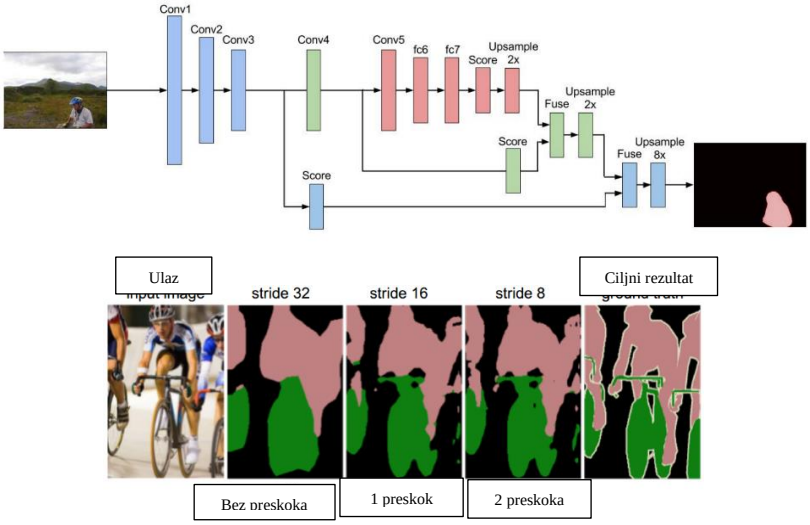
Činjenica da potpuno konvolucione neuronske mreže izbacuju potpuno povezane slojeve, kao i činjenica da operacija konvolucije nije ograničena na ulaz specifične veličine znače da ove neuronske mreže mogu da operišu nad ulaznim slikama proizvoljne veličine. Arhitektura potpuno konvolucione mreže, data Slikom 4.14, omogućava generisanje toplotne mape (eng. *heat map*), jer vrši predikciju na svakom pikselu izlazne slike. S druge strane, očigledan nedostatak ovog pristupa je u tome što sa povećanjem broja slojeva rezolucija dobijene toplotne mape klasa opada, jer svaki konvolucioni sloj, kao i svaki sloj za agregaciju, generiše manju sliku u odnosu na ulaznu sliku ili mapu obilježja na sloju.



Slika 4.14 – Arhitektura potpuno konvolucione mreže za segmentaciju slike

Pored klasične FCN mreže za segmentaciju, postoje i mnoge druge varijacije na ovu arhitekturu, koje se oslanjaju na dodatne operacije dilatirane konvolucije i transponovane konvolucije, uz korištenje preskačućih veza kako bi povećale rezoluciju izlazne slike. Ovakvi pristupi uključuju SegNet [178], DeepLab-CRF [179] i dilatirane konvolucije [180].

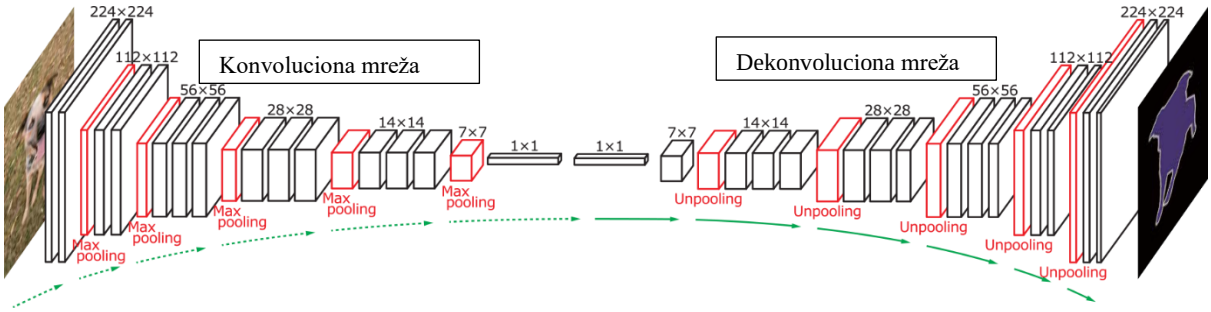
U osnovi, klasična FCN arhitektura potpuno povezane slojeve uklanja i na njihovo mjesto stavlja dodatne konvolucione slojeve i sloj za nadodmjeravanje, dok korekcije obično koriste neku vrstu preskačućih veza, kako je prikazano Slikom 4.15 [181].



Slika 4.15 – Modifikacija FCN mreže sa preskačućim vezama (gore) i poređenje rezultata sa i bez preskačućih veza (dole)

Ove preskačuće veze omogućavaju dopunjavanje ili spajanje na slojevima za nadodmjeravanje, budući da slojevi za nadodmjeravanje suštinski kopiraju ili interpoliraju informacije pri uvećavanju slike i ne unose nove informacije u rezultat. Spajanje uvećane slike sa rezultatom segmentacije iz nekog od prethodnih koraka omogućava dodavanje novih informacija u koraku nadodmjeravanja, te na taj način omogućava unošenje detaljnijih obilježja u rezultantnu sliku, kako se vidi sa Slike 4.15.

Pored ovog pristupa, koristi se i sličan pristup koji uključuje takozvani dekonvolucioni autoenkoder, odnosno, autoenkoder kod kog enkoder sadrži samo konvolucione slojeve i slojeve za agregaciju, a dekoder samo slojeve za transponovanu konvoluciju i nadodmjeravanje. Ova arhitektura, poznatija kao DeconvNet [182], prikazana je Slikom 4.16.



Slika 4.16 [182] – *DeconvNet arhitektura*

DeconvNet arhitektura eliminiše preskačuće veze, korištenjem dekonvolucionog dekodera i povećavanjem broja slojeva. Suštinski, pristup nije značajno promijenjen u odnosu na klasičnu FCN, jer i ovaj „dekonvolucioni" autoenkoder je suštinski potpuno konvolucioni autoenkoder, budući da se transponovane konvolucije mogu implementirati klasičnim konvolucijama.

Pored pomenutih arhitektura, postoje još neke, vrlo specijalizovane arhitekture, koje, opet, koriste neku varijaciju na FCN mrežu [183]. Vrijedi napomenuti i da postoji segmentaciona implementacija R-CNN mreže, poznatija kao Maskirajuća R-CNN (eng. *Mask R-CNN*) [184] koja takođe pokazuje veoma dobre performanse.

U praktičnim upotrebama, bilo koja varijanta FCN mreže obično daje dobre rezultate, pri čemu se, radi jednostavnosti, preferiraju već postojeći modeli sa već prethodno obučenim težinama koje se naknadno mogu fino podešavati.

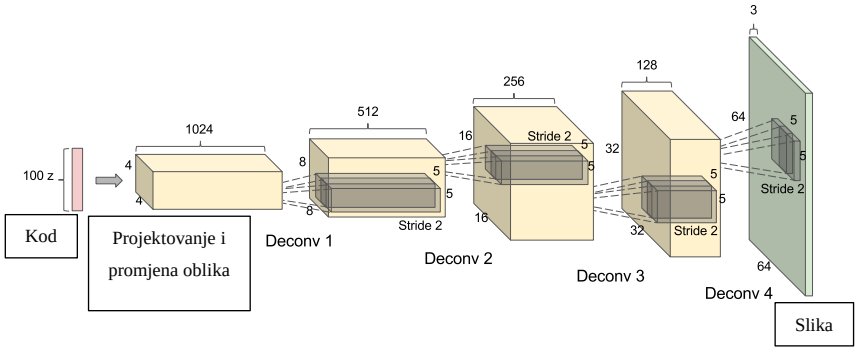
4.5 GENERISANJE SADRŽAJA

Konačno, pojavljuje se i oblast vještačke kreativnosti i mogućnost generisanja novog sadržaja na osnovu razumijevanja specifičnog domena. Kako su mreže u stanju da uče obilježja vrlo visokog nivoa, i kako su u stanju da na slikama prepoznaju vrlo kompleksne objekte i njihove interakcije, javlja se interes za iskorištavanjem interne reprezentacije ovih obilježja i generalizovano razumijevanje domena za svrhu generisanja novih primjera iz domena, koje mreža nije ranije vidjela, ali je u stanju da ekstrapolira na osnovu domenskog znanja.

Ovako generisan sadržaj može da se koristi za obučavanje drugih modela ili za direktnu primjenu u drugim oblastima gdje potencijalno može da olakša umjetnicima i dizajnerima kreativne zadatke.

Generalno, tri najpopularnija pristupa konstrukciji generativnih modela su: generativne suparničke mreže, varijacioni autoenkoderi i autoregresivni modeli, koji su kratko predstavljeni u poglavlju 2.7.

Relativno jednostavan pristup je duboka konvoluciona generativna suparnička mreža (eng. *Deep Convolutional Generative Adversarial Network*, DCGAN) [185] mreža, čija je generatorska arhitektura data Slikom 4.17. Ovdje, mreža za generisanje na ulazu dobija vektor semplovan iz uniformne raspodjele na osnovu kog, transponovanim konvolucijama i nadodmjeravanjem, na izlazu generiše sliku.

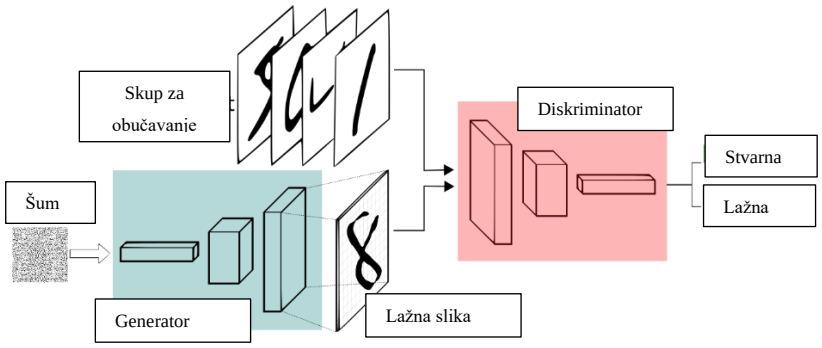


Slika 4.17 [186] – *Arhitektura generatora duboke konvolucione generativne suparničke mreže*

Budući da ne postoji jasno određeno značenje ulaznih vrijednosti, kod GAN mreža se konstruišu dvije arhitekture: generatorska mreža i diskriminatorska mreža. Ove dvije mreže rade u suparničkom režimu, gdje se diskriminatorskoj mreži na ulaz dovodi slika iz skupa za obučavanje i slika generisana generatorskom mrežom, a na izlazu očekuje indikator o tome da li je ulazna slika bila stvarna ili lažna (odnosno da li je ulazna slika dobijena iz skupa ili od generatora). Ovaj pristup omogućava propagaciju gradijenata od diskriminatorske mreže, ka generatorskoj, te na taj način obezbjeđuje da generatorska mreža generiše slike uporedive sa onim iz skupa za obučavanje, pod uslovom da se obučavanje ovih dvaju mreža ispravno stabilizovalo, te da diskriminatorska mreža, što je čest problem, nije konvergirala prije generatorske. Slika 4.18 prikazuje klasičnu kombinaciju arhitektura generatorske i diskriminatorske mreže u GAN mrežu.

Obučavanje GAN mreža obično podrazumijeva pažljivo balansiranje između performansi diskriminatora i generatora, jer u svakom trenutku jedan može da postane bolji od

drugog i na taj način konvergira u lokalni minimum. Ovo se obično događa sa diskriminatorom, jer je njegov zadatak obično lakši od zadatka generatora.



Slika 4.18 [187] – *Arhitektura GAN mreže*

S druge strane, VAE mreže omogućavaju Bajesovu inferenciju, gdje je moguće specifikovati sličnost generisanih slika, što nije slučaj kod GAN mreža. Ovo daje blagu prednost po pitanju upotrebljivosti VAE modelima za slučajeve kada su izlazne slike malih dimenzija.

Korištenjem GAN modela moguće je obučavati mreže za klasifikaciju, dodavanjem dodatnih izlaza na diskriminator, tako da diskriminator određuje i klasu ulaza. Ovo je korisno, jer se pokazuje da je moguće dobiti vrlo visoku tačnost nad daleko manjim brojem labeliranih primjera [188].

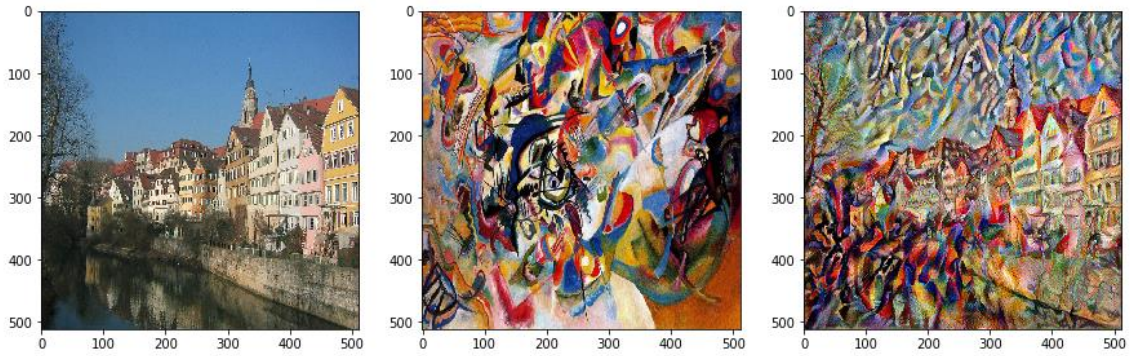
Pored pomenutog, postoje poboljšanja u generisanju kod GAN modela maksimiziranjem zajedničkih informacija za manje podskupove reprezentacionih varijabli i opservacija. Ovaj model omogućava predvidivost mapiranja iz prostora ulaza na slike, te na taj način omogućava kontrolu nad detaljima izlaza i generisanje sličnih izlaza kretanjem kroz ulazni prostor, slično kao kod dubokih konvolucionih inverznih grafičkih mreža [102], bez dodatnog nadgledanja. Ovaj pristup i arhitektura nazvani su InfoGAN [189].

Noviji GAN modeli omogućavaju i generisanje vrlo detaljnih slika visoke rezolucije (BigGAN arhitektura [190]), kao i usmjeravanje i prenos stila slika koje se generišu [191]. Rezultati usmjerenog GAN pristupa prikazani su na Slici 4.19.

Slika 4.19 [191] – *Generisane slika lica korištenjem GAN arhitekture*

Pored pomenutih primjena, GAN arhitekture se pokazuju kao vrlo praktične i za uvećavanje slike, sa dodavanjem izmišljenih (takozvanih „haluciniranih“) detalja kao oblik ekstrapolacije [192].

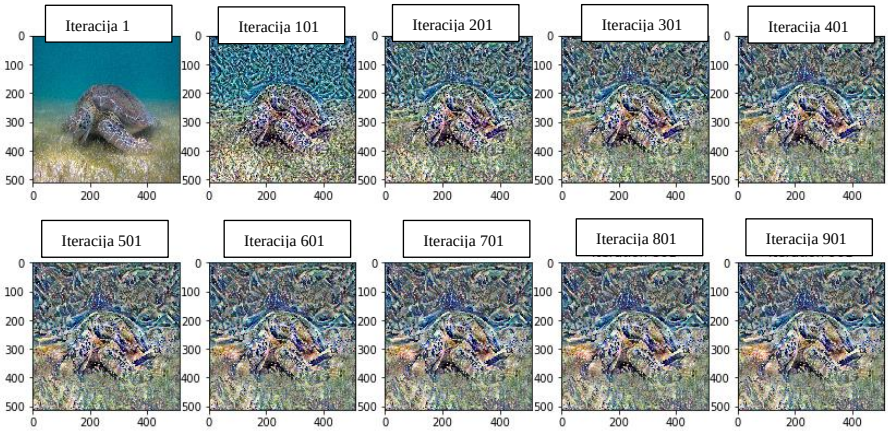
Iako se GAN arhitekture najčešće koriste pri generisanju sadržaja, pod generisanje sadržaja može da se svrsta i prenos stila, koji je postao vrlo često korištena vizuelna tehnika koja je inspirisala uvećavanje sa rekonstrukcijom detalja. Prenos stila ne uključuje GAN arhitekture, već koristi klasične arhitekture konvolucionih neuronskih mreža sa specijalizovanim algoritmom [193] koji koristi obilježja naučena na različitim slojevima proizvoljne arhitekture neuronske mreže da zamijeni prepoznata obilježja sa ulazne slike. Ovaj pristup podrazumijeva postojanje arhitekture koja je obučena na nekom skupu podataka, kako bi se stil neke slike mogao zamijeniti stilom druge, korištenjem i zamjenom najaktivnijih neurona modela za jednu i drugu sliku. Slika 4.20 prikazuje primjenu algoritma prenosa stila.



Slika 4.20 [194] – *Ulazna slika, slika sa stilom, ulazna slika sa primjenom stila, slijeva udesno*

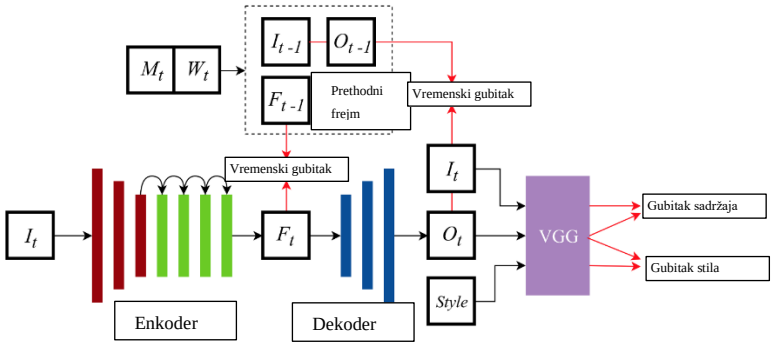
U osnovi, algoritam za prenos stila sastoji se iz nekoliko generalnih koraka. Prvo, učitava se neki model neuronske mreže sa već naučenim parametrima, obično VGG19 zbog pogodnosti rednog povezivanja slojeva i činjenice da svaki naredni sloj sadrži obilježja višeg nivoa nego kod prethodnog sloja. Nakon toga, definišu se dvije funkcije gubitka: L2 norma za funkciju gubitka za sadržaj (odnosno rastojanje između ulazne slike i ciljne slike sa sadržajem, koje se obično inicijalizuju istom slikom) i funkciju gubitka za stil (odnosno rastojanje između stilskih reprezentacija ulazne slike i slike sa stilom, gdje se stilska reprezentacija definiše kao korelacija između različitih odziva filtara za ulaznu sliku, datu kao Gramova matrica dobijena iz vektorizovanih reprezentacija mapa obilježja na svakom od slojeva) [193].

Nad ovako konstruisanim modelom pokreće se neka od varijanata gradijentskog spuštanja sa ciljem konvergencije izlaza na sliku sa prenesenim stilom. Konačna slika se dobija postepenom promjenom težina modela, kako je prikazano Slikom 4.21.



Slika 4.21 [194] – *Postepeno dobijanje slike sa prenesenim stilom*

Postoji veliki broj varijacija na ovaj pristup, uključujući temporalno stabilizovani prenos stila za prenos stila na video zapisu, kakav je, na primjer, ReCoNet [195], čija je arhitektura prikazana Slikom 4.22 [196] i super-rezolucione algoritme, koji su već prethodno navedeni.



Slika 4.22 – Protočni sistem ReCoNet arhitekture sa temporalnom funkcijom gubitka

Pored pomenutih, postoje brojne druge arhitekture i pristupi za generisanje sadržaja, ali su ovdje prezentovane najčešće korištene i većina ostalih arhitektura predstavlja neku njihovu varijaciju. Samim tim, preporuka je, ukoliko je potrebno konstruisati model za generisanje sadržaja, korištenje GAN modela ili modela za prenos stila sa već obučenim klasifikacionim modelima, najčešće onim datim u poglavlju 4.2.

5 METODE PROJEKTOVANJA I RAZVOJA NEURONSKIH MREŽA

5.1 PREGLED

Na osnovu prethodno izloženog detaljnog pregleda oblasti i preporuka formiranih na osnovu postojeće literature, ustaljenih praksi i sopstvenog istraživanja, može da se formira skup pravila, odnosno uža metodologija, za razvoj specijalizovanih neuronskih mreža za svrhe klasifikacije i segmentacije. Iako su u prethodnim glavama izloženi glavni koncepti neophodni za formiranje šire metodologije, ovdje je detaljno prezentovano nekoliko specifičnih vlastitih studija i eksperimenata, potkrijepljenih postojećom literaturom, sa ciljem detaljnije ilustracije upotrebljivih metoda i pristupa projektovanju i specijalizaciji neuronskih mreža u specifičnim oblastima i domenima upotrebe, kao i sa ciljem izdvajanja neophodnih mehanizama za replikaciju i modifikaciju rješenja prezentovanog u Glavi 6.

Neuronska mreža za klasifikaciju i segmentaciju aero-snimaka zemljišta bazirana je na izloženom znanju i principima i ova glava pobliže objašnjava te principe kako bi odluke donesene pri razvoju te neuronske mreže bile dodatno potkrijepljene i objašnjene.

U poglavlju 5.2 objašnjen je način izbora arhitekture s obzirom na problem koji se rješava. Ovdje su dodatno (u odnosu na Glavu 4) obrazloženi uslovi za izbor neke od postojećih arhitektura za rješavanje problema ili potproblema. Poglavlje 5.3 daje smjernice za konstrukciju specijalizovane arhitekture u slučajevima kada nije moguće upotrijebiti postojeći model, na primjeru vlastitih eksperimenata.

Poglavlje 5.4 bavi se izborom načina obučavanja, pristupima obučavanju i augmentaciji skupa za obučavanje, s obzirom na problem koji se rješava. Ovdje su, na osnovu ranije izloženog i uz vlastite eksperimente, date smjernice za odabir načina obučavanja specifičnih mreža.

Konačno, u poglavlju 5.5 dat je meta-algoritam koji na najvišem nivou rezimira predloženu metodologiju i predstavlja apstraktnu reprezentaciju svih smjernica navedenim u Glavi 5 i svim prethodnim glavama. Za razumijevanje koraka ovog meta-algoritma neophodni su znanje, smjernice i koncepti prezentovani u ranijim glavama. Na osnovu dijela datog meta-algoritma, projektovana je klasifikaciona mreža i njena segmentaciona varijanta, prezentovana u Glavi 6.

5.2 IZBOR ARHITEKTURE

5.2.1 Ekvivalentna algoritamska implementacija

Neki specifični problemi potencijalno predodređeni za rješavanje pristupima sa mašinskim učenjem ne moraju se nužno optimalno ili najbrže rješavati ovim pristupom. U konkretnoj vlastitoj studiji, pokazuje se da ručna selekcija konvolucionih filtara i njihova manuelna kombinacija, analogna detekciji ivica, može dati izuzetno kvalitetne rezultate, bez potrebe za obučavanjem ekvivalentne jednoslojne konvolucione neuronske mreže, primijenjeno na detekciju upala i polipa u video snimcima kolonoskopskog snimanja [24] [197].

Pokazuje se i da je, u slučaju razvrstavanja medicinskih slika koje sadrže jedan ili više grafika ili ilustracija, algoritamski pristup korištenjem konveksnog omotača efikasniji u odnosu na binarni klasifikator baziran na GoogleNet mreži, uz predobučavanje [23].

Pored toga, u seriji studija na temu detekcije malih objekata [19] [20] [21], pokazuje se da pristup korištenjem serije manuelno odabranih filtara sa eksperimentalno određenim parametrima daje ekvivalentno dobre rezultate kao i specifično konstruisana konvoluciona neuronska mreža, uz brže izvršavanje.

Ovi rezultati sugerišu na prvi, relativno očigledan, a naizgled naivan, korak u selekciji odgovarajuće arhitekture neuronske mreže: identifikacija potrebe za korištenjem mašinskog učenja pri rješavanju cijelog problema ili nekog njegovog podsegmenta. Iako ne postoji generalizovan kriterijum kada je jedan pristup bolji od drugog, moglo bi se, na osnovu iskustva, pretpostaviti da se problemi koji ne zahtijevaju razumijevanje semantike domena, kao i problemi za koje postoji jasno definisano algoritamsko rješenje ili postojeća algoritamska metoda, za koje ne postoji visok nivo šuma i varijacije na ulazu koja ne može da se prevaziđe manuelnom selekcijom filtara ili korištenjem tehnika koje se oslanjaju na obilježja niskog nivoa [3], mogu riješiti algoritamskim pristupom i ne zahtijevaju selekciju i obučavanje neuronskih mreža.

5.2.2 Podudarni domeni

Prije svega, za trivijalni slučaj kada se domenski problem suštinski svodi na već riješen problem, kao što je klasifikacija ili detekcija objekata uopštenog sadržaja, vrlo jasan pristup je selekcija neke od već postojećih arhitektura, za koje postoji vrlo veliki broj već obučenih

mreža. Štaviše, većina savremenih biblioteka za rad sa neuronskim mrežama već sadrži obučene modele nekih od klasičnih mreža datih u poglavlju 4.1.

U praksi, slučajevi u kojima je potrebna samo operacija klasifikacije su relativno rijetki, jer praktični zadaci obično zahtijevaju ili detekciju ili segmentaciju. Bez obzira na to, SSD modeli sa već predobučavanim vezama mogu da se koriste u ovim slučajevima. Kako je već naglašeno, sasvim praktičan metod, ukoliko je to neophodno, je i zamjena sadržanog VGG modela SSD mreže sa modelom obučenim na nekom drugom skupu.

Dakle, ukoliko postoji odgovarajući model koji klasifikuje u skup kategorija koje predstavljaju nadskup ciljnih kategorija, prirodan izbor je neki već postojeći model. Ukoliko postoje ograničenja po pitanju korištenih biblioteka ili alata, modeli čije su težine sačuvane u ONNX formatu (ili, praktično, u bilo kom drugom formatu) mogu relativno lako da se pretvore u modele čitljive datom alatu, za većinu modernih alata, uključujući Tensorflow [124], Caffe [51] i CNTK [198].

Budući da je SSD modele teže obučiti u odnosu na klasifikacione modele, u slučaju potrebe za detekcijom ili segmentacijom niske rezolucije koja ne zahtijeva izvršavanje u realnom vremenu, vrlo praktičan pristup je korištenje klizećeg prozora sa ili bez preklapanja, gdje se ulazna slika dijeli u rešetkastu mrežu ćelija i nad svakom ćelijom mreže vrši inferencija. Ovakvim pristupom dobija se toplotna mapa analogna onoj koja bi bila izlaz YOLO mreže ili neke FCN mreže sa niskorezolucionim izlazom. Ovaj pristup je značajno sporiji od inferencije YOLO mreže, ali se pokazuje kao vrlo efikasan u praksi, te ne zahtijeva dodatno obučavanje mreže za ovu namjenu.

Štaviše, postoje alati koji omogućavaju praktičnu implementaciju mreža deklarativnim jezikom, kakav je slučaj sa Caffe alatom, gdje nije potrebno dodatno znanje nekog specifičnog programskog jezika za konstrukciju i obučavanje upotrebljive mreže. Za slučajeve inferencije, postoje alati, kao što je Digits [199] alat za rad sa neuronskim mrežama, koji omogućavaju izvršavanje inferencionih zadataka na hostovanom serveru i konstruišu API (eng. *Application Programming Interface*) na nivou HTTP protokola (eng. *Hypertext Transfer Protocol*) za komunikaciju sa serverom i obavljanje udaljenih zadataka inferencije. Ovakav pristup olakšava rad sa neuronskim mrežama u proizvoljnom programskom jeziku.

Na konkrentom vlastitom primjeru, korištenje pripremljenog SSD modela sa VGG mrežom se pokazalo kao praktično upotrebljivo u slučaju detekcije i klasifikacije stojećih vozila na četvorotračnoj cesti, gdje je korišteni model bio obučen nad ImageNet skupom i bez

dodatnog obučavanja upotrebljen za detekciju specifičnih kategorija vozila, isključivanjem aktivacija za klase koje nisu od interesa, uz postizanje performansi od nekoliko frejmova u sekundi. Ovdje je detekcija mirujućih vozila izvršena algoritamski, detekcijom preklapanja obuhvatajućih pravougaonika u susjednim frejmovima, gdje se sva preklapanja ispod podešenog praga smatraju istim vozilom, a sva ostala poklapanja, zavisno od smjera trake, različitim vozilima ili vozilima u pokretu. Integracija neuronske mreže u ovaj sistem zahtijevala je minimalne adaptacije, gdje je primarna bila konverzija težina između različitih formata i umetanje težina u SSD mrežu. Suštinski, ukoliko postoji adekvatan model, dovoljno sličan svojim izlaznim podacima traženom domenu, njegova upotreba se preferira u odnosu na razvijanje specijalizovanog modela, osim u slučajevima kada postoje specifična domenska ograničenja.

Slika 5.1 prikazuje rezultat segmentacije sa klizećim prozorom bez preklapanja dobijen korištenjem Digits alata i specijalizovane neuronske mreže za klasifikaciju [1]. Ovaj pristup pokazao se kao vrlo efikasan na slučajevima kod kojih su ulazne slike rezolucije 5000×5000 piksela, gdje je označavanje po slici trajalo ne više od 32 sekunde. Korišteni model je bio specifično obučavan GoogleNet model. Dakle, prethodno obučen model za klasifikaciju korišten je u cjelini i bez izmjena za rješavanje zadatka segmentacije[11].

[11] Korištena arhitektura za klasifikaciju, njeno obučavanje i adaptacija za zadatak segmentacije, koja je glavna tema ove disertacije, biće detaljno razrađena u Glavi 6.

Slika 5.1 – Segmentacija klasifikacionim modelom korištenjem klizećeg prozora

S druge strane, često je moguće iskoristiti i manje složene arhitekture mreža za rješavanje sličnih zadataka ili, ukoliko postoji odgovarajući skup za obučavanje, obučiti postojeće dublje arhitekture smanjujući broj parametara po sloju, konkretno broj filtara. Ovo se pokazuje kao efikasan način uštede memorijskih i izračunskih resursa. Vrijedi napomenuti da promjena veličine kernela, kao i promjena broja i strukture slojeva podrazumijeva promjenu ili adaptaciju arhitekture.

5.2.3 Adaptacija postojećih arhitektura

U slučajevima kada ne postoji adekvatna arhitektura neuronske mreže za dati domenski problem, adaptacija neke postojeće arhitekture koja se pokazala kao efikasna u rješavanju sličnog problema je prirodan korak.

Kako je ranije rečeno, često ne postoji dovoljno velik skup za obučavanje da bi se duboke arhitekture kao što su Inception, VGG ili ResNet obučile, pa je neophodno pronaći način da se postojeće težine, naučene na sličnom problemu, prenesu u novu arhitekturu.

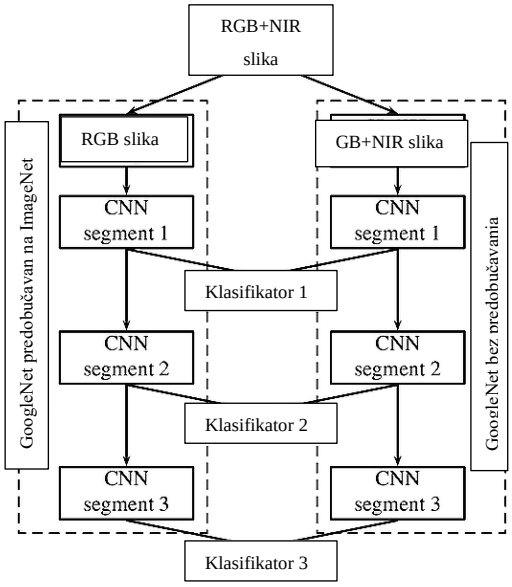
Ovdje su se kao efikasna, u eksperimentima, pokazala dva pristupa: korištenje poznate manje arhitekture, kao što je AlexNet, i njeno postepeno proširivanje i produbljivanje, zavisno od dobijenih eksperimentalnih rezultata, i kombinovanje nekoliko postojećih arhitektura u jednu, čuvajući parametre dobijene obučavanjem na srodnom skupu. Ovdje vrijedi naglasiti da srodan problem ne mora nužno biti sličan na dubljem nivou, ali može dijeliti obilježja niskog nivoa i na taj način doprinijeti tačnosti dodatno obučavane mreže.

U slučaju postepenog proširivanja arhitekture, osnova pristupa je u čuvanju parametara plitkih slojeva, koji već sadrže mape obilježja niskog nivoa, dok se parametri dubljih slojeva mogu mijenjati. Različiti alati dozvoljavaju postavljanje koeficijenata za skaliranje koeficijenta učenja na nivou individualnih slojeva, kao što je slučaj sa Caffe alatom. Pored toga, dodatne slojeve ima smisla dodavati uglavnom samo iza slojeva čiji se parametri čuvaju iz prethodno obučene mreže. Parametre dodatnih slojeva najefikasnije je inicijalizovati Xavier ili He inicijalizacijom, iz ranije diskutovanih razloga. Pri proširivanju AlexNet arhitekture, ukoliko se dodaju novi konvolucioni slojevi, pokazuje se kao efikasno dodavati i slojeve za agregaciju, pri čemu je važno voditi računa o smanjenju veličine izlaza na svakom sloju, ukoliko se ne koristi dopunjavanje. Suštinski, dodavanje novih slojeva zaustavlja se kada tačnost na validacionom skupu prestane da se povećava, a tipično se povećava sa dodavanjem novih slojeva. Broj kernela po sloju se generalno, po uzoru na većinu poznatih arhitektura i prema izvedenim eksperimentima iz pomenutih vlastitih studija, povećava sa dubinom mreže, kulminirajući u drugoj polovini mreže, posmatrano od ulaza. Ovaj broj kernela rijetko prelazi 1024 i obično se kreće oko 256, ali se generalno preferira povećavanje dubine mreže u odnosu na broj kernela po konvolucionom sloju, gdje se dobri rezultati mogu postići i sa znatno manjim brojem kernela, što ubrzava performanse mreže. Naravno, kako dubina mreže raste, tipično raste i broj kernela po sloju, pa tako neke od često korištenih arhitektura sadrže veći broj kernela po sloju nego što je ovdje navedeno.

Modifikacija arhitekture ne mora nužno da bude samo zamjena dubokih slojeva. Na primjer, u slučaju klasifikacije višekanalnih slika, odnosno multispektralnih slika, ulaz mreže nije očekivani trokanalni ulaz za koji postoje već obučeni modeli. U slučaju odabira postojeće arhitekture neuronske mreže, ne bi bilo moguće direktno iskoristiti naučene parametre, jer ulaz nije iste dimenzionalnosti. Pored toga, ukoliko su klase različite, neophodno je promijeniti sve potpuno povezane slojeve, ukoliko oni postoje u odabranoj arhitekturi. Dakle, u ovakvim slučajevima, dodatno obučavanje je neizbježno, osim u slučaju da se informacije u dodatnim kanalima mogu ignorisati bez gubitka tačnosti, što je relativno malo vjerovatno.

Jedno potencijalno rješenje je kombinovanje dvaju ili više postojećih arhitektura u jedinstvenu arhitekturu, čuvajući naučene parametre za postojeće arhitekture, uz dodatno obučavanje. U konkretnoj vlastitoj studiji [18], dvije Inception mreže kombinovane su u jednu mrežu podijeljenu na dvije grane, slično kao AlexNet mreža (iako je AlexNet mreža podijeljena u dvije grane zbog memorijskih ograničenja), s ciljem klasifikacije multispektralnih slika scena (pejzaža). Kako postoje već obučeni modeli Inception mreže na ImageNet skupu, koji uključuje i slike pejzaža, ovaj model je odabran kao polazni. Budući da multispektralne slike, u ovom slučaju, sadrže četiri kanala, eksperimentisano je sa nekoliko različitih konfiguracija mreže.

Prije svega, testirane su performanse predobučavane Inception mreže sa izbacivanjem jednog od četiri kanala, kako bi se predobučavane težine mogle iskoristiti. Taj trivijalni pristup upoređen je sa pomenutom kombinacijom mreža, čija je arhitektura data Slikom 5.2.



Slika 5.2 – Kombinovana dualna mreža kod koje su segmenti ekvivalentni onima u GoogleNet mreži

Kako se vidi sa slike, jedna od mreža sadržavala je težine obučene na ImageNet skupu, dok je druga mreža inicijalizovana nasumično, Xavier inicijalizacijom. Budući da Inception mreža sadrži tri funkcije gubitka na tri različita klasifikatora, spajanje mreža izvedeno je na ovim klasifikatorima, pri čemu su te težine obučavane sa novim vrijednostima, zbog promjene dimenzionalnosti izlaza, odnosno promjene traženog broja klasa. Četvorokanalni ulaz se, odgovarajućim slojevima za rutiranje, dijelio na RGB kanale za prvu mrežu (jer ona sadrži obučena obilježja nad slikama sa ova tri kanala) i GB i infracrveni kanal (eng. *near infra-red,* NIR) za drugu mrežu.

Dati pristup kombinovanju pokazuje daleko bolje rezultate od svih prethodno postignutih nad istim skupom podataka, kako je prikazano Tabelom 5.1. Tabela 5.1 takođe prikazuje nekoliko eksperimenata koji su izvedeni sa jednom GoogleNet arhitekturom. Iz tabele se vidi da konstruisana dualna mreža sa rutiranjem iskorištava informacije iz sva četiri kanala, jer pokazuje značajno bolje rezultate od korištenja jedne trokanalne arhitekture.

TABELA 5.1 [18] – POREĐENJE PRETHODNOG NAJBOLJEG PRISTUPA SA PREDLOŽENIM

Algoritam	Tačnost
Prethodni najbolji (FV+SVM, PCA RGB + SIFT [200] [18])	87.9%
GoogleNet RB+NIR	90.3%
GoogleNet RG+NIR	90.3%
GoogleNet RGB	83.3%
Dualna mreža sa svim kanalima	**92.5%**

Ovdje prva kolona predstavlja način kombinovanja različitih kanala. Pored toga, rezultati su upoređeni sa dotadašnjim pristupima, uz ranu fuziju kanala, gdje se pokazuje da je ovaj pristup kombinovanju arhitektura značajno efikasniji od prethodnih.

Vrijedi napomenuti da problemi koji zahtijevaju klasifikaciju ili detekciju objekata ili neke specifične vrste objekata predstavljaju srodan problem klasifikaciji nad ImageNet skupom, dok neki drugi, vrlo specijalizovani, problemi, kao što je klasifikacija rendgenskih snimaka, zahtijevaju obučavanje bez predobučavanja, jer su obilježja niskog nivoa suštinski različita između ovih dvaju domena, pa ne postoji značajan transfer znanja iz jedne oblasti u drugu. Ovaj problem, kako se pokazalo, se najjednostavnije rješava odabirom odgovarajuće nasumične inicijalizacije parametara slojeva, obično Xavier ili He, kako je ranije naglašeno.

Mreže je takođe moguće adaptirati postepenim izmjenama neke postojeće arhitekture. Moguće je spajanje dubljih slojeva neke odabrane, prethodno obučene arhitekture, sa plitkim slojevima neke druge postojeće arhitekture, uz dodatno obučavanje. Alternativno, vrlo čest postupak je i korištenje postojećih arhitektura, uz izdvajanje nekih slojeva ili dodavanje novih. U takvim slučajevima, plići dio mreže se često koristi kao izvor vektora obilježja za druge klasifikatore ili mreže. Postupak razmjene slojeva ili modula između mreža nerijetko se naziva *transplantacija*. Kao dodatna ilustracija ove ideje služi i primjer generativne suparničke mreže za generisanje rukom pisanih cifara. Za svrhe vlastite studije koja je uključivala razvoj pristupa stabilizaciji dubljih GAN mreža [201], bilo je potrebno konstruisati relativno dubok generatorski model, za koji bi odgovarajući diskriminator bila modifikovana LeNet mreža. Ovdje je, za diskriminator, LeNet arhitektura modifikovana izuzimanjem slojeva za agregaciju, dok je generatorska mreža konstruisana obrtanjem redoslijeda slojeva diskriminatora i zamjenom operacije konvolucije sa transponovanom konvolucijom, analogno autoenkoderskom ili prorijeđenom autoenkoderskom pristupu. Ovako konstruisana mreža pokazala se kao efikasna u generisanju rukom pisanih cifara, bez konvergencije ka nekoj specifičnoj klasi. Sličan pristup inkrementalnoj adaptaciji GAN mreža iskorišten je i u drugim studijama dostupnim u literaturi [202].

Sličan model korišten je i u odvojenom eksperimentu [201], gdje je mreža predobučavana kao diskriminator u GAN modelu, da bi naknadno bila obučavana, sa naučenim parametrima kao inicijalizacionim vrijednostima, nad MNIST [203] skupom. Uz dodatne modifikacije u načinu obučavanja, ova arhitektura postigla je tačnost od 99.9% nad ovim skupom, što je iznad dotadašnjeg najboljeg ostvarenog rezultata. Iako ovakav rezultat ne predstavlja značajan pomak, budući da se smatra da je ovaj skup savladan, govori u prilog funkcionalnosti modifikacionog pristupa razvoju arhitektura.

Dodatan pristup koji se u praksi često pokazuje kao vrlo praktičan je upotreba takozvanih ansambala mreža, gdje se nekoliko različitih mreža odvojeno obučava, te se njihov zajednički rezultat, na neki način normalizovan, koristi kao konačan rezultat. U praktičnoj studiji [22], korišten je veliki ansambl veoma plitkih neuronskih mreža za postizanje vrlo dobrih rezultata u klasifikaciji rukom pisanih cifara. Ovdje su individualni klasifikatori obučavani binarno nad klasama, sa ciljem specijalizacije za pojedinačne klase, a njihovi binarni izlazi spojeni u vektor vjerovatnoće pripadnosti klasama. Pored toga, diferencijalni klasifikatori, za svaki par klasa, obučeni su da razlikuju klase, u slučajevima kada se na izlaznom vektoru pojavljuju nedovoljno

separabilne vrijednosti. Ovaj pristup, iako je koristio veoma plitke individualne klasifikatore, davao je rezultate uporedive sa dotadašnjim najboljim, odnosno 99.2%.

Postojeće publikacije na temu adaptacije arhitektura neuronskih mreža za zadatke različite od onih za koje su arhitekture inicijalno obučavane podržava prethodno izloženi materijal. Kada se radi o kombinaciji različitih arhitektura, vrlo često se koriste naučena obilježja nižeg nivoa iz neke od postojećih, prethodno obučenih, arhitektura, pa se na plitke slojeve takvih mreža nadovezuju dodatni slojevi. Ova pristup naziva se stekovanje (eng. *stacking*) [204] i oslanja se na činjenicu da su obilježja niskog nivoa relativno konzistentna između različitih zadataka [205] [206] [207].

Ukoliko se posmatra prethodno prezentovana literatura, većina najčešće korištenih arhitektura neuronskih mreža predstavljala je neku modifikaciju ili nadogradnju prethodno publikovanih arhitektura, sa generalnom tendencijom povećavanja dubine, a smanjivanja parametara po sloju, kao i korištenja učenja sa transferom, uključujući MTL i predobučavanje. Kako su najefikasnije arhitekture, kao što su Inception, ResNet i dijelom VGG, izgrađene od individualnih modula, pristup modifikaciji ovih arhitektura uveliko je pojednostavljen, jer se može svesti na izbacivanje, dodavanje i zamjenu modula unutar jedne arhitekture, ili kombinovanjem modula između različitih arhitektura. Za konstrukciju dubokih mreža sa relativno velikim brojem modula, pokazuje se da korištenje preskačućih veza značajno smanjuje probleme sa nestajućim gradijentom i omogućava dalje povećanje tačnosti sa povećanjem dubine. U većini praktičnih slučajeva, za oblasti koje su razumno bliske originalnim, ovakav pristup pokazuje se kao najbrži za razvoj sa malom vjerovatnoćom neuspješne konstrukcije.

5.3 KONSTRUKCIJA SPECIJALIZOVANE ARHITEKTURE

Iako komparativno rijetki, slučajevi koji iziskuju konstrukciju arhitekture mreže specifične za neki problem predstavljaju najkompleksniji oblik problema, u kojima je relativno teško formulisati generalizovanu metodologiju, budući da su ovi slučajevi obično izuzetno usko specijalizovani ivični slučajevi. Slučajevi upotrebe kada je potrebno konstruisati specijalizovane arhitekture uglavnom podrazumijevaju istraživanje i razvoj, pa je često konstrukcija novih mreža upravo otvoren problem u oblasti mašinskog učenja.

Ipak, postoji nekoliko uopštenih pravila koja mogu da se iskoriste radi povećavanja vjerovatnoće uspjeha. Ovdje su date smjernice za konstrukciju arhitekture mreže čije se

obučavanje fundamentalno oslanja na stohastičko gradijentsko spuštanje ili neku njegovu varijantu. Oblasti koje uključuju učenje sa podsticajem ili nenadgledano učenje ne podliježu nužno ovim pravilima, odnosno ne postoji dovoljno akumuliranih empiričkih dokaza za validnu generalizaciju pravila konstrukcije.

Prije svega, neophodno je identifikovati tip problema i vrstu podataka koji postoje za svrhu obučavanja. Ovdje se podrazumijeva da je moguće obezbijediti skup za obučavanje odgovarajuće veličine, bilo ručnim označavanjem, augmentacijom ili automatskim generisanjem.

Nakon obezbjeđivanja odgovarajućeg skupa, prirodan korak je manuelna inspekcija skupa s ciljem formulisanja apstraktne algoritamske specifikacije rješenja datog problema. U poglavlju 5.2 rečeno je da izbor između algoritamskog pristupa i pristupa korištenjem mašinskog učenja suštinski zavisi od nivoa šuma prisutnog na nivou podataka nad kojim se operiše. Ukoliko je odabran pristup mašinskim učenjem, podrazumijevano postoji šum na nivou ulaznih podataka za koji se od mreže zahtijeva kompenzacija. Pokazuje se, na osnovu brojnih vlastitih eksperimenata, kao i na osnovu onih iz literature, da arhitektura mreže obično mora da oslikava apstraktne korake neophodne za kompenzaciju na uočeni šum, pri čemu šum ne podrazumijeva nužno raznovrsnost obilježja niskog nivoa, već postojanje kompleksnih varijacija na nivou obilježja višeg nivoa. Ovo podrazumijeva i potencijalno kodovanje otpornosti na rotacione i translacione operacije, kao i operacije skaliranja.

Apstraktno gledano, neuronska mreža svojom arhitekturom (slojevima i njihovim povezivanjem) predstavlja seriju apstraktnih koraka uopštenog algoritma za rješenje datog problema, gdje se operacije na individualnim koracima uče za vrijeme obučavanja.

Dodatno, kako je diskutovano u ranijim poglavljima, neophodno je izbjeći sve potencijalne probleme pri konstrukciji, uključujući nestanak i eksploziju gradijenata, overfiting i konvergenciju u lokalni minimum. Ovo se uopšteno rješava izborom odgovarajućih aktivacionih funkcija, korištenjem preskačućih veza, regularizacijom, učenjem sa transferom i odgovarajućim podešavanjem hiperparametara za vrijeme obučavanja, na načine koji su ranije objašnjeni.

Pokazuje se da korištenje učenja sa transferom daleko poboljšava performanse mreža, pa se ovaj pristup preferira ukoliko uslovi to dozvoljavaju, čak i ako postoji veliki broj uzoraka za obučavanje za ciljni zadatak. Ovo često podrazumijeva definisanje različitih funkcija gubitaka i neophodno je naglasiti da ove funkcije ne moraju nužno biti usmjerene ka istom cilju (na

primjer, kod GAN mreža, dvije definisane funkcije gubitka imaju suštinski različite ciljeve). U opštem slučaju, postojaće funkcija gubitka koja će indukovati pristrasnost mreže ka rješavanju traženog problema, ali će, uz nju, postojati dodatne funkcije gubitka sa ciljem omogućavanja generalizacije. Na osnovu predstavljenih vlastitih studija, kao i predstavljenih studija dostupnih u literaturi, može se izvesti zaključak da konstrukcija dodatnih funkcija gubitaka u većini slučajeva treba da bude takva da se optimizuju parametri individualnih koraka projektovanog apstraktnog algoritma. Odnosno, dodatne funkcije gubitka imaju za cilj da indukuju pristrasnost mreže ka rješavanju zadatka na način predviđen pri konstrukciji arhitekture. Dakle, arhitektura mreže odgovara specifikaciji apstraktnog algoritma, za slučaj kada bi individualni koraci sadržavali idealne parametre, odnosno bili u stanju da optimalno riješe ili izvedu apstrahovani korak algoritma, dok minimizacija funkcija gubitka ima za cilj da omogući konvergenciju parametara ka pretpostavljenim idealnim vrijednostima. Ovo osigurava da su dobijeni rezultati uvijek rezultat projektovanja mreže, a ne slučajnog pogađanja upotrebljive arhitekture. Iako je algoritamska konstrukcija arhitekture izvodljiva [208], dobijene arhitekture obično mogu da se predstave i opišu kao sekvencijalni koraci nekog apstraktnog algoritma [209] [210] [211] [212].

Prema tome, radi ekvivalencije semantičkog modela rješenja problema sa arhitekturom mreže kao algoritamskog rješenja, trebalo bi da mreža zadovolji sljedeća tri uslova:

- → Arhitektura mreže oslikava generalne korake koji bi se mogli formulisati pri konstrukciji algoritamskog rješenja, podrazumijevajući da postoje idealni parametri na svakom od koraka (grupa uzastopnih slojeva bez grananja); projektovanjem apstraktnog algoritma za rješavanje datog problema, projektuje se i arhitektura mreže, gdje svaki modul ili segment mreže odgovara nekom koraku algoritma, pri čemu se uzastopni slojevi dodaju u module radi povećavanja dubine mreže sa ciljem povećavanja kompleksnosti koncepata koje mreža može da nauči dok god se tačnost mreže povećava
- → Arhitektura mreže omogućava propagaciju gradijenata za svaku od definisanih funkcija gubitaka na takav način da gradijenti ne divergiraju i da se svaka od funkcija optimizuje ka zadovoljavajućem ekstremumu
- → Funkcije gubitka su definisane tako da se parametri podešavaju sa ciljem optimizacije definisanih apstraktnih koraka projektovanog algoritma, a ne nužno samo sa ciljem dobijanja očekivanih izlaza za dati problem

Pored navedenog, vrijedi napomenuti da dostupnost adekvatnog skupa za obučavanje može značajno da utiče na izbor pristupa obučavanju. Na primjer, nedovoljan broj uzoraka za obučavanje često iziskuje neki vid proširivanja skupa za obučavanje augmentacijom ili generisanjem novih uzoraka. Vrlo često se, u nedostatku uzoraka za obučavanje, koristi i MTL. Dostupnost hardvera za obučavanje i vremenska i resursna ograničenja takođe mogu da utiču na izbor pristupa obučavanju, pa se algoritamski razvoj uveliko oslanja na dostupnost resursa za obučavanje.

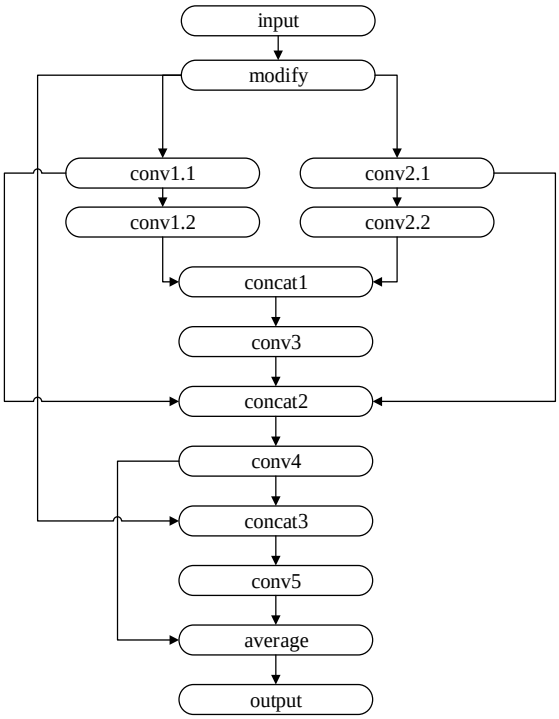
Ovaj algoritamski ekvivalentan pristup vrlo dobro može da se ilustruje na konkretnoj studiji čiji je cilj bila detekcija i segmentacija malih objekata u pokretu [19]. Ovdje je konkretan zadatak bio detekcija insekata u pokretu na video zapisima visoke rezolucije. Glavni problem u ovoj vrsti detekcije, nakon rješavanja problema nedostatka anotiranog skupa za obučavanje korištenjem proceduralne generacije testnih parova[12], je činjenica da su objekti od interesa izuzetno mali i zamućeni kao rezultat njihovog kretanja relativno velikom brzinom, poređeno sa frekvencijom odmjeravanja kamere. Na individualnim frejmovima video zapisa, golim okom ovi objekti su gotovo potpuno neprimjetni.

Na osnovu prethodno izloženih klasičnih modela, ovdje se lako zaključuje da korištenje nekog postojećeg modela nije moguće, jer objekti koje je neophodno detektovati ne predstavljaju entitete slične onima nad kojima se mreže klasično obučavaju, a, dodatno, objekti su dovoljno maleni (svega nekoliko piksela u promjeru), da klasični SSD modeli nemaju odgovarajuću prostornu rezoluciju za detekciju. Prirodan izbor ovdje je konstrukcija neke vrste FCN mreže, budući da je zadatak fundamentalno segmentacioni.

S obzirom na to da se insekti dovoljno brzo kreću da rijetko postoje na istom mjestu u ekranskom prostoru na dva susjedna frejma, prihvatljivo je pretpostaviti da bi se pri projektovanju specijalizovanog algoritma vršilo neko poređenje susjednih frejmova [20] [21]. Samim tim, pri konstrukciji ekvivalentne neuronske mreže, neophodno je indukovati pristrasnost mreže ka diferenciranju susjednih frejmova, bilo izborom slojeva, povezivanjem ili izborom odgovarajuće funkcije gubitka. Očigledno, ovakva mreža morala bi prihvatati susjedne frejmove na ulazu, bilo konkatenirane po širini ili dužini ili po kanalima. U implementiranom slučaju, konkatenacija je rađena po kanalima, kako bi se indukovala

[12] Ovaj pristup augmentaciji objašnjen je detaljno u poglavlju 5.4.3.

pristrasnost mreže ka traženju razlike između susjednih frejmova. Slika 5.3 prikazuje arhitekturu projektovane mreže.



Slika 5.3 – Mreža za detekciju malih objekata u pokretu

Kako se vidi sa slike, koristi se FCN mreža sa preskačućim vezama i grananjem. Grananje, na slici označeno kao *modify* sloj, konstruisano je korištenjem slojeva za oduzimanje, koji su konstruisali tri nova kanala, na osnovu razlike odgovarajućih kanala za prvi i drugi frejm. Grananje postoji kako bi se indukovala djelomična invarijantnost na skaliranje (jer se detektovani insekti potencijalno kreću prema i od kamere), korištenjem kernela različitih veličina, kako je prikazano Tabelom 5.2.

TABELA 5.2 – MODULI KONSTRUISANE FCN MREŽE

Modul	Parametri slojeva
conv1.1	16, 3, ReLU 32, 5, ReLU 64, 7, ReLU 64, 5, ReLU
conv1.2	32, 5, ReLU 16, 3, ReLU
conv2.1	16, 5, ReLU

	32, 7, ReLU 64, 9, ReLU 64, 7, ReLU
conv2.2	16, 3, ReLU
conv3	32, 3, ReLU 32, 5, ReLU 24, 3, ReLU
conv4	32, 5 ReLU 16, 3, ReLU 1, 3, Sigmoid
conv5 (conv1.1 + conv1.2)	16, 3, ReLU 32, 5, ReLU 64, 7, ReLU 64, 5, ReLU 32, 5, ReLU 16, 3, ReLU 1, 3, Sigmoid

Odabir potpuno konvolucione mreže za ovu svrhu bio je uslovljen činjenicom da mreža mora da generiše masku koja označava insekte na individualnim frejmovima, te zbog potencijalne promjenljivosti veličine ulaznih frejmova. Ovo takođe omogućava izvršavanje u segmentima, u slučaju distribucije na hardver sa ograničenim resursima.

Sloj za računanje prosječne vrijednosti ovdje ima za svrhu dijeljenje gradijenata, tako da se dio mreže specijalizuje za detekciju objekata, a dio za uklanjanje artefakata izazvanim kretanjem i potresima kamere. Inicijalno, mreža je konstruisana i obučena bez ovog koraka, a on je dodat, na osnovu opservacije tačnosti, zbog pojave artefakata kretanja. Dodatni koraci omogućavaju mreži da uči kako da uklanja ove artefakte. Alternativno, mogle su biti definisane dvije funkcije gubitka, bez korištenja sloja za računanje prosječne vrijednosti (na *conv4* i *conv5* modulima). Ovdje je arhitekturom određena pristrasnost mreže ka uklanjanju ovih artefakata. Dobijeni rezultati prikazani su Slikom 5.4.

Slika 5.4 – Ulazna slika (samo prvi frejm) i odgovarajuća maska na izlazu za mrežu za detekciju malih objekata u pokretu

Ovaj pristup, kako je ranije navedeno, pokazuje se kao ekvivalentno efikasan manuelnom pristupu, pri čemu je robusniji na promjene okruženja, jer se može iznova obučavati ponuđenom polunadgledanom metodom.

S druge strane, u velikom broju specifičnih primjena, kada je neophodno projektovati i obučiti novu mrežu, pomenute mreže ne moraju biti veoma duboke, niti razgranate. Ovo je posebno relevantno u slučajevima regresije nad sekvencama podataka.

Na konkretnoj vlastitoj studiji, gdje je neuronska mreža korištena u IoT (eng. *Internet of Things*) kontekstu za predviđanje vrijednosti senzora IoT mreže u slučaju njihovog otkaza [25], može se ilustrovati činjenica da je neophodna dubina mreže, pogotovo u slučaju FFNN modela, relativno mala. U slučaju FFNN modela, bez konvolucionih slojeva, obično je dovoljno konstruisati mrežu sa dva do tri potpuno povezana skrivena sloja, gdje su češće varijante sa manjim brojem slojeva. Ovo je potkrijepljeno literaturom, kako je navedeno u Glavi 2 i Glavi 3.

Budući da je u pitanju regresioni model čiji su ulazi vremenska serija, ovdje se može diskutovati ranije pomenuto razmotavanje rekurentnih mreža. Umjesto korištenja rekurentnih jedinica, u ovom eksperimentu u memoriji je čuvana istorija prethodnih n odmjeraka za svaki senzor i pri svakoj inferenciji mreže na ulaz su dovođene sve vrijednosti iz tog prozora. Dakle,

ulaz je formiran kao dvodimenziona matrica $n \times m$ u kojoj su za svaki od m senzora dati odmjerci iz posljednjih n trenutaka u vremenu. Ovakvo formiranje ulaza svodi konstrukciju rekurentnog prediktora na klasičnu konvolucionu neuronsku mrežu. U eksperimentima se pokazalo da je za zadovoljavajuću predikciju ovdje dovoljno iskoristiti klasičnu FFNN mrežu, bez potrebe za konvolucijama i razmotavanjem. Pored toga, ova mreža je obučavana aktivno, izvršavanjem koraka propagacije unazad svaki put kada nove vrijednosti senzora postanu dostupne. Ovako predikcija vremenom postaje bolja za neprekidan tog ulaznih podataka.

Serijom odvojenih eksperimenata, inspirisanom Inception arhitekturom, ustanovljeno je da konvolucioni slojevi, dodati iza potpuno povezanih slojeva, mogu povećati tačnost modela, kao rezultat činjenice da je izvedena operacija rijetka (odnosno, ne definiše parametre nad svim kombinacijama ulaza).

Problem konstruisanja specijalizovane arhitekture može se svesti na nekoliko generalnih principa. Prije svega, ukoliko se ne uočava posebna kompleksnost koja bi zahtijevala specifikovanje apstraktnog algoritma, odnosno, ne uočava se potreba za razgranavanjem apstraktnog algoritma za svrhu mapiranja ulaza u izlaze, dati problem se može realizovati plitkom neuronskom mrežom, bez grananja. Dodavanje novih slojeva vrši se od izlaznih slojeva, ka ulazu, pri čemu broj potpuno povezanih slojeva ne prelazi tri. Ukoliko je neophodno dodavanje slojeva preko ovog broja, dodaju se konvolucioni slojevi (potencijalno, uz slojeve za agregaciju), pri čemu je poželjno, ali ne neophodno, ulazne podatke urediti tako da operacija konvolucije predstavlja smislen korak u odgovarajućem algoritmu, odnosno postoji eksplicitno objašnjenje značenja kombinacije ulaza na dati način. Iako ulazni podaci ne moraju nužno biti ovako strukturisani da bi se dovodili na ulaze konvolucionih slojeva, u slučaju da nisu, operacija konvolucije može da sakrije međusobnu zavisnost podataka koji bi eventualno postojali na ulazu, jer kombinuje podatke na osnovu mapa obilježja koje su obično značajno manjih dimenzija u odnosu na cijeli ulazni tenzor. U ovom slučaju, ako postoji dovoljna hardverska podrška (odnosno ne postoje značajna memorijska ograničenja), odgovarajuća preskačuća veza ili jednoslojno grananje često omogućava iskorištenje obe operacije, jer obezbjeđuje da se informacije koje bi inače bile izgubljene ili sakrivene proslijede dublje u mrežu. U gotovo svim klasifikacionim eksperimentima, pogotovo kada su u pitanju slike, dovođenje ulaza velikih dimenzija na potpuno povezane slojeve ne rezultuje dobrim performansama. Ovdje se preferira korištenje konvolucionih slojeva i slojeva za agregaciju za redukciju dimenzionalnosti ulaza prije slanja potpuno povezanim slojevima ili slojevima za globalnu prosječnu agregaciju.

U slučajevima kada je potrebno konstruisati dublju arhitekturu, često su od koristi već ustanovljeni moduli (kao što je slučaj sa Inception ili ResNet mrežama), pa se umjesto dodavanja slojeva, koriste ovako razgranati moduli koji se kao cjelina umeću u mrežu umjesto individualnog sloja. Grane mreže prikazane na Slici 5.3 ilustruju korištenje jednog modula inspirisanog Inception modulom. Pomenute dvije grane mogu se smatrati modulima, jer same koduju obilježja različitih veličina, pa predstavljaju mrežu u mreži.

Konačno, bilo koji segment već postojeće mreže može se iskoristiti kao modul mreže, tako da konstrukcija specijalizovanih mreža rijetko podrazumijeva razvoj potpuno nove arhitekture. Obično se konstrukcija arhitekture svodi na kombinovanje manjih modula (bilo direktno preuzetih, modifikovanih ili napravljenih po uzoru) iz najčešće korištenih mreža. S druge strane, ovdje se javljaju dodatni problemi koji uključuju način obučavanja projektovanih arhitektura, jer se, kako je već rečeno, arhitekture moraju konstruisati tako da budu prikladne za predviđeni način obučavanja.

S obzirom na to da konstrukcija nove specijalizovane arhitekture tipično podrazumijeva i neki vid istraživanja, vrlo vjerovatno ne postoji jednostavan metodološki pristup koji bi omogućio rješavanje bilo kog problema na ovaj način. Prethodno izloženi pristup koji se oslanja na definisanje koraka apstraktnog algoritma će davati dobre rezultate u onoj mjeri u kojoj projektant rješenja može da sagleda prirodu problema i modeluje ga kao algoritamsku protočnu strukturu, kroz detaljno razumijevanje oblasti od interesa i ispravnu metakognitivnu introspekciju u proces koji dovodi do rješavanja datog zadatka od strane projektanta. Drugim riječima, metodološki je moguće konstruisati specijalizovanu arhitekturu neuronske mreže samo ukoliko projektant razumije i oblast od interesa i prirodu problema i u stanju je da apstraktno modeluje algoritamski proces koji preslikava očekivane ulaze u očekivane izlaze. Ukoliko projektant nije u stanju da izvede ovo modelovanje na adekvatnom nivou, tada se izlazi iz okvira metodološkog rješavanja problema, i problem postaje istraživačke prirode.

5.4 IZBOR NAČINA OBUČAVANJA

5.4.1 Hiperparametri

Iako je već bilo riječi o izboru i podešavanju hiperparametara, ovdje će biti dato nekoliko primjera koji ilustruju bitnije pristupe. Kako postoji veliki broj hiperparametara koji bi se mogli uzeti u obzir, ovdje su dati primjeri koji ulustruju podešavanje onih čije podešavanje

najpouzdanije poboljšava rezultate, te će na osnovu tih primjera biti izvedeno nekoliko pravila i preporuka.

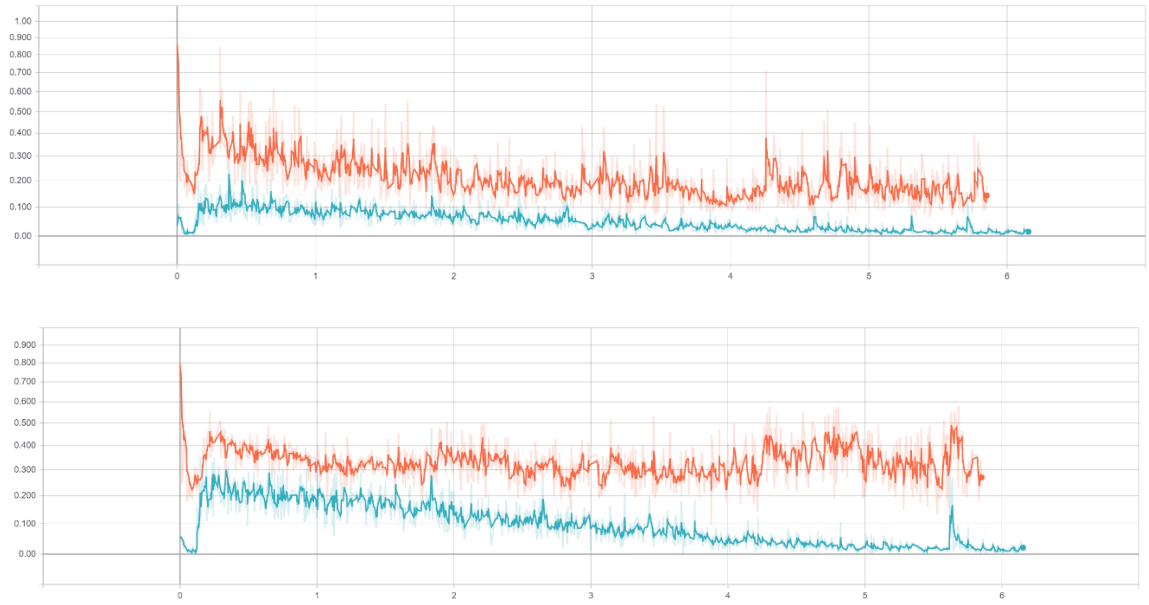
Napomenuto je da je koeficijent učenja jedan od najbitnijih hiperparametara i da generalno veoma značajno utiče na rezultate obučavanja. Takođe je pomenuto da konvergencija mreže uveliko zavisi od optimizatora koji se koristi, pa, shodno tome, ove hiperparametre je potrebno dodatno obraditi.

Suštinski, pri obučavanju se formuliše način promjene koeficijenta učenja zavisno od broja iteracije ili broja epohe, gdje se počinje sa nekom relativno velikom vrijednošću (na primjer 0.01, ukoliko je veličina grupe dovoljno velika, odnosno bar 256) koja se postepeno smanjuje sa brojem iteracije ili epohe. Najčešće korišten pristup je stepenasto smanjivanje množenjem koeficijenta učenja nekim faktorom (obično u rasponu između 0.5 i 0.99, zavisno od problema) na svakih n proteklih epoha, gdje se vrijednost n obično kreće između 1 i 10, ponovo zavisno od problema i trajanja obučavanja. Vrlo često se ovi hiperparametri određuju na osnovu nekoliko sesija obučavanja i poređenja tačnosti na validacionom skupu, ali se, u principu, cilja na to da se koeficijent učenja atenuira od početne vrijednosti do neke upotrebljive vrijednosti u posljednjoj epohi, budući da nije smisleno koristiti izuzetno niske vrijednosti koeficijenta učenja, jer parametri neće biti adekvatno ažurirani. Razlog za atenuaciju je fino podešavanje parametara, odnosno dozvoljavanje konvergencije u najbliži minimum.

Pored ovog pristupa, kao vrlo efikasan se pokazao i pristup sa dvofaznim obučavanjem [1] [18], gdje se mreža prvo obuči korištenjem jednog optimizatora, do konvergencije ili sa ranim zaustavljanjem, a potom se koeficijent učenja ponovo podesi na nešto veću vrijednost (ne na početnu) i nastavi se obučavanje korištenjem drugačijeg optimizatora, sa opadajućim koeficijentom učenja, do konvergencije. Tipično, ovaj pristup popravlja tačnost za nekoliko procenata, jer omogućava mreži da konvergira u minimum blizak potencijalno pronađenom od strane prvog optimizatora. Preveliki skokovi u promjeni koeficijenta učenja mogu dovesti do divergencije mreže, iako je moguće koeficijent učenja postaviti na inicijalnu vrijednost u drugoj fazi. Ovaj postupak se obično naziva restartovanje [213].

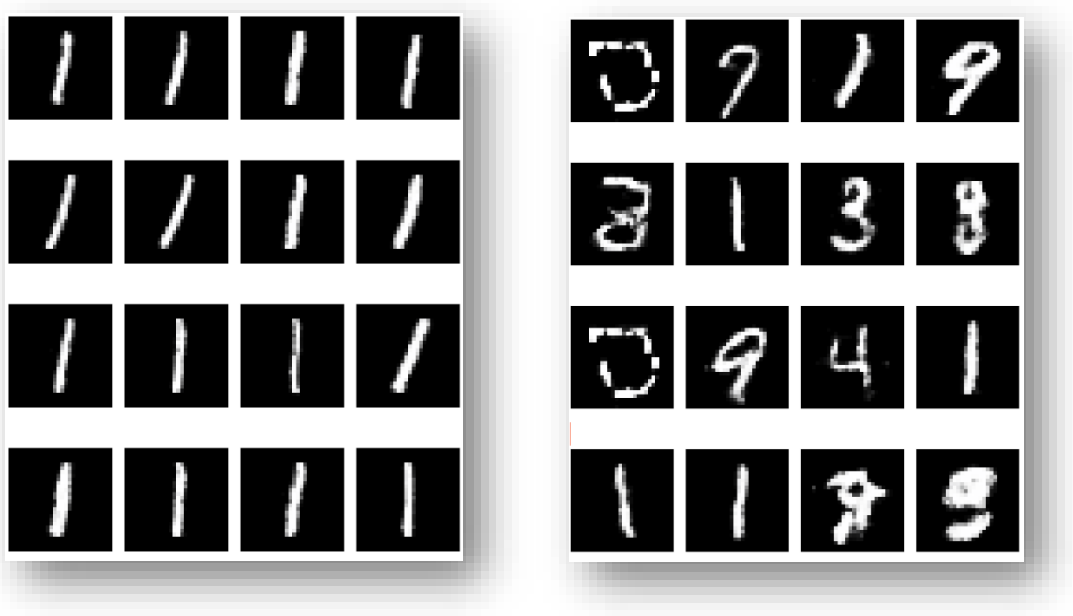
S druge strane, u odvojenoj vlastitoj studiji, pokazalo se da se podešavanjem koeficijenta učenja s ciljem usporavanja učenja mreže može značajno doprinijeti generalizaciji u slučaju stabilizacije GAN mreža [201]. Poznato je da je najveći problem pri obučavanju generativnih suparničkih mreža njihova međusobna stabilizacija i omogućavanje balansiranog

napretka u učenju. Drugim riječima, potrebno je obezbijediti da generatorska i diskriminatorska mreža uče približno jednakom brzinom, kako bi se izbjeglo nadjačavanje (obično u prilog diskriminatora). Ovdje su koeficijenti učenja podešavani u svakoj iteraciji obučavanja na osnovu odnosa vrijednosti funkcija gubitaka za generator i diskriminator. Koeficijenti učenja za generatorski i diskriminatorski (Adam) optimizator podešavani su na takav način da njihov odnos inverzno proporcionalno odgovara odnosu između vrijednosti funkcije gubitka za generatorsku i diskriminatorsku mrežu. Dakle, kada jedna mreža postane bolja, koeficijent učenja njenog optimizatora se smanjuje i obratno. Ovdje je bitno primijetiti da se, iako se radi o Adam optimizatoru koji tipično ne zahtijeva podešavanje koeficijenta učenja, performanse mreže povećavaju. Slika 5.5 prikazuje odnos vrijednosti funkcija gubitka za slučaj sa i bez stabilizacije koeficijenata učenja, dok Slika 5.6 prikazuje rezultate generatora bez i sa stabilizacijom.



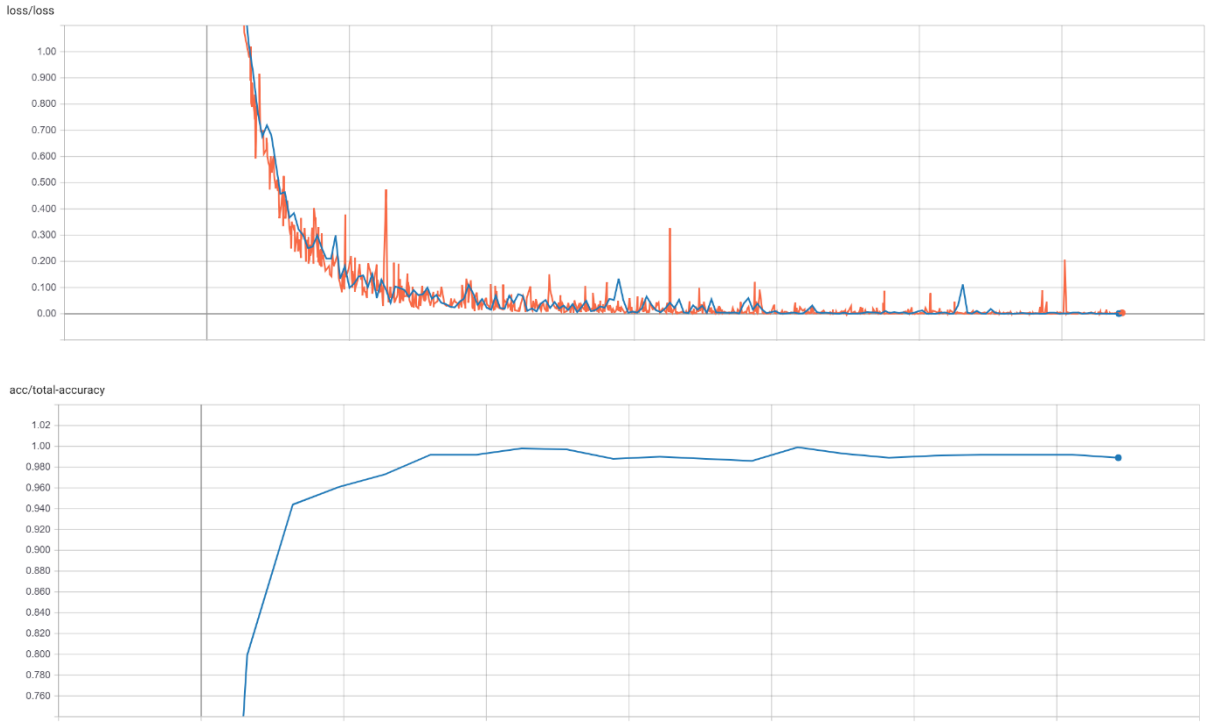
Slika 5.5 – Funkcija gubitka za generatorsku mrežu (gore) i funkcija gubitka za diskriminatorsku mrežu (dole), za slučajeve sa i bez stabilizacije (slučaj bez stabilizacije prikazan je tamnijom nijansom)

Naizgled, moglo bi se pretpostaviti da je slučaj sa stabilizacijom dao lošije rezultate, jer je vrijednost funkcije gubitka veća, ali to nije slučaj, jer mreža ne konvergira u lokalni minimum, kako se vidi sa Slike 5.6. Iz slike se jasno vidi da mreža bez stabilizacije konvergira u minimum gdje generator proizvodi samo one slike kojima je najjednostavnije prevariti diskriminator (u ovom slučaju slike cifre 1). S druge strane, sa stabilizacijom se dobijaju raznovrsniji rezultati.



Slika 5.6 – Razlika u izlazima generatora bez (lijevo) i sa (desno) stabilizacijom, nakon istog broja iteracija

Dodatno, korištenje stepenasto opadajućeg koeficijenta učenja, polovljenjem koeficijenta učenja na svakih 5000 iteracija bilo je ključni faktor u savladavanju MNIST skupa sa 99.9% tačnosti. Slika 5.7 prikazuje vrijednosti funkcije gubitka i tačnost mreže kroz broj iteracija.



Slika 5.7 – Funkcija gubitka (gore; narandžasto – gubitak na skupu za obučavanje, plavo – gubitak na skupu za validaciju) i tačnost mreže (dole), u odnosu na broj iteracija, do 20000 iteracija

U slučajevima odabira koeficijenata optimizatora, najčešće su optimalni parametri sugerisani od strane autora, pa prema tome, individualne koeficijente, pored koeficijenata učenja, obično nije potrebno podešavati. Ovdje vrijedi napomenuti da se broj kernela po slojevima može povećavati sa ciljem povećanja tačnosti mreže, pri čemu se obično preferira dodavanje mehanizma (ili slojeva, zavisno od biblioteke) za odbacivanje, kako bi se izbjeglo pojavljivanje podudarnih kernela, odnosno specijalizacija različitih kernela za isti zadatak.

Pored savjeta datih u poglavlju 3.7, moguće je korištenje i genetičkog algoritma za pronalazak optimalnih hiperparametara, ukoliko je mreža relativno mala i vrijeme obučavanja sa jednim skupom hiperparametara relativno kratko. Ovdje je neophodno naglasiti da izbor skale pri nasumičnom određivanju hiperparametara značajno doprinosi trajanju odabira hiperparametara, jer očekivana raspodjela vjerovatnoće boljih vrijednosti hiperparametara, kao što je koeficijent učenja, za odabrani raspon pretrage, ne mora nužno biti linearna. S druge strane, u praktičnim testovima se pokazalo da podučeno pogađanje obično daje prihvatljive rezultate brže nego korištenje algoritma koji radi pretragu i ponovno obučavanje, bilo da se radi o diskretnom iteriranju kroz prostor hiperparametara ili o pretrazi genetičkim algoritmom.

Prema tome, za optimizaciju procesa obučavanja, najčešće je neophodno postepeno smanjivati koeficijent učenja (često i u slučajevima kada se koristi neki od optimizatora za koji takvo nešto teoretski nije neophodno). Ostali hiperparametri mogu da se biraju ili algoritamski (ukoliko postoji dovoljno raspoloživo vrijeme i resurzi za obučavanje) ili edukovanim pogađanjem na osnovu problema. Koji god pristup da se odabere, najčešće je svaki hiperparametar moguće optimizovati nezavisno od ostalih.

5.4.2 Učenje sa transferom

Učenje sa transferom je vjerovatno jedna od najznačajnijih tehnika pri obučavanju neuronskih mreža, jer se u testovima pokazala, pored skaliranja koeficijenta učenja, kao pristup koji najviše doprinosi poboljšanju tačnosti.

Učenje sa transferom podrazumijeva nekoliko različitih pristupa, dominantno MTL pristup koji mrežu ili obučava za više zadataka simultano ili finim podešavanjem (eng. *fine tuning*) parametara prenesenih sa nekog drugog zadatka. Ranije je pomenuto da se pri konstrukciji neuronskih mreža preferira korištenje modula iz poznatih arhitektura neuronskih mreža. Ovo takođe daje mogućnost kopiranja naučenih parametara za te specifične module.

U prethodno pomenutim studijama [1] i [18], ovaj pristup korišten je ili za prenos parametara cijele mreže, gdje je mreža obučavana sa dva različita optimizatora u dvije faze, ili

za prenos parametara dijela mreže, gdje je samo jedna grana mreže sadržavala prenesene težine, što se, u tom slučaju, pokazalo kao bolje od prenošenja obučenih parametara na obe grane (jer su dvije ugniježdene Inception mreže ovako sadržavale različite inicijalizacije).

Pored toga, pri finom podešavanju prenesenih parametara, moguće je skalirati koeficijent učenja različito na različitim slojevima. Ovdje se kao najefikasniji pristup pokazuje zamrzavanje svih ili dijela parametara na plitkim slojevima, a postepeno povećavanje koeficijenta učenja sa povećavanjem dubine. Ovo se radi sa pretpostavkom da su naučena obilježja niskog nivoa obično slična, bez obzira na klasu, pa je rijetko potrebno ove parametre obučavati ponovo i rizikovati njihovu divergenciju.

Kako je ranije rečeno, vid učenja sa transferom je i pristup sa predobučavanjem mreže na drugom problemu, gdje se mreža obučava u dva koraka: prvo obučavanje izvodi se na sličnom problemu, a drugo nad problemom koji se rješava. Primjer ovakvog pristupa je u eksperimentu klasifikacije aero-snimaka [1], eksperimentu sa proširenim skupom (odnosno većim brojem klasa)[13], kao i kod pristupa sa dualnom mrežom [18].

Dodatno, obučavanje iste mreže u različitim režimima takođe se može smatrati učenjem sa transferom, kao što je slučaj obučavanja mreže u ulozi diskriminatora u GAN arhitekturi, prije transplantacije u modifikovanu arhitekturu sa svrhom klasifikacije. Ovaj pristup omogućava inicijalizaciju mreže na sličnom problemu, gdje generatorska mreža služi kao mehanizam augmentacije i proširivanja ulaznog skupa podataka, te na taj način obezbjeđuje generalizaciju i izbjegavanje overfitinga.

Obično se pri transplantaciji mreža i u slučajevima kombinovanja arhitektura i njihovoj adaptaciji za različite probleme neki slojevi moraju ponovo obučavati. Ovo je skoro uvijek slučaj sa dubljim slojevima, a uvijek slučaj sa izlaznim potpuno povezanim slojevima, kada se mijenja domen problema. Štaviše, i u slučajevima da je broj klasa isti, ove slojeve bi trebalo ponovo inicijalizovati, jer se originalna semantika klasa, koja je kodovana u ovim slojevima, skoro sigurno promijenila. Ovo važi i u slučaju globalne prosječne agregacije.

5.4.3 Augmentacija

Augmentacija ulaznih podataka predstavlja vrlo često korišten oblik pretprocesiranja koji podrazumijeva povećavanje broja ulaznih podataka, odnosno labeliranih parova, sa ciljem poboljšanja tačnosti i generalizacije mreže.

[13] Eksperiment nad proširenim skupom urađen je za svrhe ove disertacije i opisan je u poglavlju 6.6.

Tipičan primjer je promjena kontrasta na ulazu, bez obzira na tip podataka, skaliranje amplitude ulaza, kao i dodavanje šuma. U slučajevima kada se operiše nad slikama, moguće su promjene kontrasta, varijacije u boji i svjetlini kao i dodatno filtriranje ulaza nekim predefinisanim algoritmima, kao što je zamućivanje, izoštravanje i slično.

U slučajevima kada je neophodno kodovati rotacionu invarijantnost, manuelno rotiranje ulaznih slika se pokazuje kao jedan od najefikasnijih pristupa. Analogni pristup važi i za invarijantnost na skaliranje, smicanje i slično.

Pored toga, na konkretnom, vlastitom, primjeru klasifikacije aero-snimaka [1], moguće je konstruisati ili modifikovati mrežu da prihvata ulazne slike manjih dimenzija u odnosu na one koje postoje u skupu za obučavanje (u konkretnom slučaju, modifikovana Inception mreža prihvata slike dimenzija 224×224, dok su slike u skupu 256×256). Augmentacija je ovdje izvedena nasumičnim isijecanjem slika odgovarajućih dimenzija iz ulaznih slika, kako bi se ulazni skup višestruko povećao. U nadovezanoj studiji, pokazuje se da je moguće generisati rotacije za proizvoljan ugao isijecanjem iz originalne slike, rotacijom i skaliranjem, te da ovakav pristup daje bolje rezultate (ovo je slučaj kada orijentacija ulaza nije bitna).

Kao ekstremni primjer augmentacije služi eksperiment sa polunadgledanim učenjem kod detekcije malih objekata u pokretu [19], gdje je cijeli skup za obučavanje generisan na osnovu nekoliko primjera izolovanih slika pčela. Ovdje je obučavanje rađeno aktivno, proceduralnim generisanjem slika u svakoj iteraciji obučavanja. Drugim riječima, cijeli skup za obučavanje algoritamski je generisan na osnovu svega nekoliko slika pčela izolovanih od pozadine i video zapisa prazne pozadine. Na osnovu video zapisa pozadine i izolovane slike insekta generisan je par ulaz-izlaz, preklapanjem sa pozadinom i direkcionim zamućivanjem izolovane pčele u proizvoljnom smjeru. Ovakav pristup generisao je imitaciju slike sa insektom na ulazu i odgovarajuću masku na izlazu. Ovdje je bitno napomenuti da je generisanje rađeno tako da su ulazne slike sadržavale relativno ujednačen odnos dijela slike sa objektom i dijela slike bez objekta, kako bi se spriječila konvergencija mreže u lokalni minimum, indukovanjem pristrasnosti ka praznoj detekciji. Kako se radi o FCN mreži, veličina ulaza nije morala biti fiksna, pa su za obučavanje korištene slike onih dimenzija koje dozvoljavaju adekvatno balansiranje broja piksela koji sadrže pčelu i piksela koji sadrže pozadinu. Dodatno, korištena je binarna maska (crno-bijela maska, umjesto nijansi sive), kako bi se omogućila propagacija numerički većih vrijednosti gradijenata.

U većini slučajeva kada se radi o relativno kompleksnom problemu, preferira se neki pristup augmentaciji ulaznih podataka, bilo naivnim metodama varijacije ulaza ili kompleksnijim generisanjem novih slika ili lažnih slika, uključujući i korištenje generativnih modela.

5.4.4 Pretprocesiranje, postprocesiranje i međuprocesiranje

Tipično, postprocesiranje se radi sa ciljem uklanjanja šuma, izglačavanja izlazne slike ili dodatnom algoritamskom segmentacijom. U slučaju detekcije malih objekata, korak postprocesiranja iskorišten je u obliku primjene Gausovog zamućenja i odsijecanja sa pragom, koji je imao za cilj uklanjanje preostalog šuma koji mreža nije bila u stanju da otkloni, kao i naglašavanje detektovane maske za dalje korake detekcije lokacije.

Klasičan primjer pretprocesiranja je normalizacija ulaznih podataka u odnosu na prosječnu vrijednost za cijeli skup za obučavanje (pri čemu se ista normalizacija, sa istom dobijenom prosječnom vrijednošću, koristi i pri inferenciji, odnosno testiranju), kako je bio slučaj i u implementiranom pristupu klasifikaciji aero-snimaka.

Naravno, nije praktično enumerisati sve moguće načine pretprocesiranja i postprocesiranja, jer oni suštinski zavise od konkretnog problema koji se rješava. Ono što je bitno naglasiti i što se može zaključiti na osnovu široko dostupne literature, kao i iskustva u projektovanju specijalizovanih algoritama [23] [214] [215], je da se određeni koraci u rješavanju problema bolje mogu riješiti korištenjem mašinskog učenja, dok za druge postoje odgovarajući algoritamski pristupi. Na ovaj način se pretprocesiranje može smatrati algoritamskom transformacijom ulaznih podataka na putu do neuronske mreže, dok se postprocesiranje tumači kao algoritamska transformacija izlaza mreže sa ciljem rješenja problema. Budući da je samo projektovanje arhitekture neuronske mreže predstavljeno kao definisanje apstraktnog algoritma, procesi pretprocesiranja i postprocesiranja mogu se smatrati dijelom cijele arhitekture, gdje određeni segmenti sadrže parametre koji se uče, a drugi parametre koji se specifikuju izvornim kodom. Drugim riječima, cijeli algoritam rješavanja problema sastoji se od kodiranih segmenata (za koje postoji izvorni kod) i obučavanih segmenata (za koje postoji arhitektura neuronske mreže i njeni parametri), pa se tako korak postprocesiranja za jedan obučavani segment može smatrati korakom pretprocesiranja za sljedeći obučavani segment, odnosno, ovi koraci zajedno predstavljaju svojevrsno „međuprocesiranje“ podataka između dvaju obučavanih segmenata.

Na ovaj način, pri projektovanju rješenja problema koje potencijalno uključuje pristup mašinskim učenjem, problem se ne posmatra kao projektovanje arhitekture neuronske mreže sa algoritamskim pretprocesiranjem i postprocesiranjem, već se algoritamski koraci povezuju sa modulima koji sadrže neuronske mreže u jednu koherentnu cjelinu koja predstavlja algoritam za rješenje cjelokupnog problema. Ovakav način posmatranja rješavanja problema indukuje pristrasnost projektantu ka odabiru algoritamskih koraka za segmente rješenja koja su pogodna za rješavanje algoritamski, kao i odabiru obučavanih koraka prikladnih pristupu mašinskim učenjem. Suštinski, ovo proširuje ideju projektovanja arhitekture neuronske mreže kao algoritma, na projektovanje algoritma koji sadrži obučavane parametre na različitim koracima, u kom slučaju se sloj generalizuje na korak odgovarajućeg algoritma, pa tako sloj ne mora da sadrži parametre koji se obučavaju.

5.5 META-ALGORITAM

Konačno i na najvišem nivou, uzimajući u obzir sve izloženo i bez specifičnih detalja za koje su smjernice i metode već date, pristup projektovanju i razvoju specijalizovanih arhitektura neuronskih mreža može, u opštem slučaju, da se svede na sljedeći meta-algoritam.

→ Identifikovati korake apstraktnog algoritma koji rješava dati problem[14], sa specifikacijom individualnih koraka koji bi se potencijalno mogli riješiti bilo algoritamski ili korištenjem nekog inferencionog modela neuronske mreže, podrazumijevajući idealizovane vrijednosti parametara i bez raščlanjivanja individualnih komponenti mreža.

→ Ukoliko postoji potreba za koracima koji zahtijevaju korištenje neuronske mreže, ponoviti prvi korak, podrazumijevajući korištenje već obučene neuronske mreže, bez dodatnog obučavanja.

→ Ukoliko ne postoji odgovarajuće rješenje koje integriše postojeću obučenu mrežu (ili mreže), ponoviti prvi korak, podrazumijevajući korištenje postojeće arhitekture neuronske mreže, uz učenje sa transferom i minimalnu modifikaciju modela.

→ Ukoliko ne postoji odgovarajuće rješenje koje koristi postojeću arhitekturu, identifikovati mogućnost redefinisanja formulacije problema, s ciljem svođenja na korištenje postojeće arhitekture, bilo promjenom organizacije ulaza mreže ili načina akvizicije ulaznih podataka.

[14] Podrazumijeva se faza analize problema koja prethodi fazi projektovanja, pa nije data kao izričit korak.

→ Ukoliko ne postoji odgovarajuće rješenje koje koristi postojeću arhitekturu, identifikovati korake apstraktnog algoritma koji rješava dati problem, segmentirajući cijeli algoritam na serije koraka sa i bez obučavanih parametara, preferirajući grupisanje koraka sa obučavanim parametrima i minimiziranje broja ovih grupa (efektivno, minimizirajući broj različitih neuronskih mreža), uz raščlanjivanje individualnih komponenti korištenih mreža.

→ Ukoliko je nađeno potencijalno rješenje, testirati sve elemente pristupa koji zahtijevaju obučavanje parametara, obučavanjem odgovarajućih mreža nad dostupnim podacima.

→ Ukoliko postojeći podaci za obučavanje nisu dovoljno obimni, proširiti skup augmentacijom ili redefinisati strukturu algoritma kako bi se ulazni podaci promijenili, odnosno kako bi se koristio drugačiji skup podataka za rješavanje istog zadatka.

→ Ukoliko postojeći podaci za obučavanje i dalje nisu dovoljno obimni, redefinisati strukturu mreže (ili mreža) da koristi učenje sa transferom iz sličnih domena i koristiti podatke iz dodatih domena.

→ Ukoliko mreža (ili mreže) ne daje dovoljno dobre rezultate, a procjenjuje se da problem zahtijeva detekciju obilježja dubljeg nivoa, povećavati dubinu mreže, na predložene načine, dok god se tačnost na validacionim skupovima povećava.

→ Ukoliko mreže i dalje ne daju dovoljno dobre rezultate, podesiti hiperparametre na predloženi način.

→ U slučaju dobijanja nezadovoljavajućih rezultata, pokušati redefiniciju algoritma i ponoviti prethodno navedene korake, uz razmatranje alternativnih pristupa obučavanju (pored propagacije unazad) ili redefiniciju modela neurona[15]. Ovaj slučaj sugeriše usku specijalizaciju i problem potencijalno nepogodan za rješavanje sugerisanim pristupom inferencionim mrežama.

Dati meta-algoritam predstavlja apstrakciju pristupa razvoju specijalizovanih neuronskih mreža i predstavljen je u preglednom obliku s ciljem rezimiranja svih prethodno navedenih tehnika i metoda. Individualni koraci u razvoju će uvijek varirati s obzirom na dobijeni problem, ali brojni eksperimenti i akumulirano iskustvo pokazuju da se, u većini slučajeva, rezultujući proces projektovanja i razvoja svodi na navedeni skup koraka. Devijacije od

[15] Redefinisanje modela neurona prikazano je u Glavi 2, na primjeru LSTM neurona, i dodatno obrađeno sa aspekta specijalnih slojeva i načina njihovog definisanja u različitim alatima, u Glavi 3.

ovakvog pristupa su vrlo rijetke i neophodne su samo u slučajevima prethodno neistraženih oblasti, odnosno u slučajevima kada je neophodno riješiti neki od trenutno neistraženih problema. Ukoliko se rješava takav, otvoren, problem, generalizovana metodologija nije smislena, i predmet je aktivne diskusije [216]. S druge strane, za veliki broj praktičnih, potencijalno komercijalnih, svrha, ovaj pristup bi trebalo u najkraćem vremenu da rezultuje konstrukcijom upotrebljivog rješenja koje integriše arhitekturu vještačke neuronske mreže.

6 KLASIFIKACIJA I SEGMENTACIJA AERO-SNIMAKA ZEMLJIŠTA

6.1 PREGLED

Na osnovu izložene metodologije, u ovoj glavi je predstavljeno rješenje za klasifikaciju i segmentaciju aero-snimaka korištenjem specijalizovane arhitekture konvolucione neuronske mreže. Ovdje je detaljno opisana arhitektura ove neuronske mreže, način njenog obučavanja, kao i njene performanse u konkretnom problemu koji se rješava. Konstrukcija arhitekture ove mreže kao i proces njenog obučavanja pratili su izloženu metodologiju, te je ta metodologija u značajnoj mjeri potvrđena ovom studijom.

Poglavlje 6.2 pobliže opisuje oblast od interesa u odnosu na Glavu 1, te detaljnije predstavlja problem i relevantna istraživanja iz polja, od kojih su neka iskorištena u narednim poglavljima radi poređenja performansi i rezultata.

Poglavlje 6.3 opisuje postavke eksperimenata iskorištenih za validaciju funkcionalnosti modela mreže za svrhu klasifikacije i segmentacije, te izlaže iskorištene skupove podataka.

U poglavlju 6.4 opisan je predloženi pristup za klasifikaciju i segmentaciju aero-snimaka zemljišta, uključujući arhitekturu mreže i način njenog obučavanja.

Na osnovu eksperimenata predstavljenih u poglavlju 6.3 i pristupa opisanog u poglavlju 6.4, u poglavlju 6.5 prikazane su performanse mreže i upoređene sa prethodnim pristupima.

Konačno, poglavlje 6.6 prikazuje rješenje za veći broj klasa sa rezultatima specifičnim za ovu disertaciju, kao dopunu relevantnoj studiji [1]. Ovdje su prikazani dodatni eksperimenti, njihovi rezultati i upoređena tačnost mreže sa postojećim rezultatima iz literature. Ova proširena studija takođe služi kao validacija upotrebne vrijednosti izložene metodologije i pristupa obučavanju specijalizovane neuronske mreže za klasifikaciju aero-snimaka predstavljenog u ovoj glavi.

6.2 MOTIVACIJA I POTREBA ZA AUTOMATSKOM KLASIFIKACIJOM

U praksi, najčešće korišteni pristupi oslanjali su se na opis slika zemljišta na osnovu obilježja niskog nivoa, često semantički predstavljajući slike zemljišta kao teksture, za svrhu klasifikacije [3]. Bilo da se radi o izvlačenju obilježja na osnovu boje, histogramom ili

drugačije, na osnovu većeg broja kanala kod multispektralnih slika ili različitim predefinisanim deskriptorima niskog nivoa, čak i u slučajevima kada ove metode daju relativno dobre rezultate, i dalje postoji nedostatak više semantičke reprezentacije sadržaja ovih slika, odnosno nedostatak korištenja obilježja visokog nivoa i izvođenja dubljih koncepata. Korištenjem konvolucionih neuronskih mreža i dubokog učenja, ove fundamentalne mane su dijelom prevaziđene, uz uvođenje dodatnih izračunskih zahtjeva i potrebe za akvizicijom velikih skupova podataka za obučavanje.

Uz adekvatne resurse i početne uslove, pristupi sa deskriptorima obilježja niskog nivoa, kao i pristupi sa neuronskim mrežama daju praktično upotrebljive rezultate, ali se, zbog mogućnosti generalizacije i razumijevanja obilježja visokog nivoa obično preferira pristup sa neuronskim mrežama, pri čemu je bitno naglasiti da je uz adekvatno podešavanje moguće slične efekte dobiti i korištenjem različitih pristupa kombinovanjem obilježja niskog nivoa [217]. Najčešće korišteni pristupi bazirani na deskriptorima obilježja niskog nivoa uključuju ekstrahovanje obilježja niskog nivoa predefinisanim algoritmima, postprocesiranje obilježja kao i statističke klasifikatore (iako postoje kompleksniji načini korištenja obilježja niskog nivoa [217]). Glavna prednost ovakvih pristupa je mogućnost učenja obilježja na osnovu relativno malih skupova za obučavanje, posebno u slučajevima korištenja deskriptora niske dimenzionalnosti. S druge strane, kako je ranije detaljno razrađeno, duboke neuronske mreže, posebno konvolucione neuronske mreže, sekvencijalno kombinuju obilježja s ciljem dobijanja veoma duboke semantičke reprezentacije sadržaja, što se pokazuje kao najbolji pristup za ovu vrstu problema, posebno kada se koriste mreže predobučavane na sličnom skupu [4].

Za svrhu ovog istraživanja implementirana je neuronska mreža obučavana u dva koraka, uz predobučavanje, koja je korištena za rješavanje zadataka klasifikacije i segmentacije (odnosno detekcije regiona od interesa). Korišten je UCMerced [4] skup podataka, te su poređene performanse prethodnih najboljih implementacija sa predloženom. Dodatno, inspirisano prethodnim istraživanjima [6], mreža za klasifikaciju i algoritam za segmentaciju i detekciju primijenjeni su na skupu podataka Geološkog Prikupljanja u SAD-u (eng. *U.S. Geological Survey,* USGS) [7].

Ovaj pristup implementiran je i validiran unutar jednog dvosedmičnog perioda, što ilustruje validnost predložene metodologije. Dio metoda izloženih u Glavi 5 je direktan rezultat predstavljenog rješenja. Suštinski, predloženi pristup [1] predstavlja studiju slučaja upotrebe predložene metodologije u oblasti klasifikacije aero-snimaka.

6.2.1 Relevantna istraživanja

Pristupi bazirani na obilježjima niskog nivoa obično se oslanjaju na lokalna obilježja prvog ili drugog nivoa i obično su konstruisani tako da iskoriste lokalne teksturalne osobine sa različitim oblicima invarijantnosti (obično invarijantnost na skaliranje, translaciju i rotaciju, a, u slučaju aero-snimaka, potencijalno i na orijentaciju, odnosno ugao snimka). U slučaju višekanalnih ulaza, odnosno slika u boji, svaki kanal se obično tretira nezavisno, odnosno detekcija obilježja se vrši po kanalu, a spajanje se radi kao naredni korak. Vrlo efikasni pristupi ove vrste uključuju transformaciju obilježja invarijantnu na skaliranje (eng. *scale-invariant feature transform*, SIFT) [218], kao i histogram orijentisanih gradijenata [219]. Lokalna obilježja se često koriste zajedno sa grupisanjem vizuelnih riječi (eng. *bag of visual words*) kako bi se izračunali konačni deskriptori koji predstavljaju histogram pojavljivanja različitih lokalnih obilježja detektovanih na datoj slici [8] i ovo se pokazuje kao najefikasniji pristup kada se koriste obilježja niskog nivoa [220]. Pored toga, deskriptivne karakteristike lokalnih obilježja niskog nivoa mogu se efikasnije koristiti kombinacijom njihovih prostornih osobina, tako da se lokalna obilježja uzimaju u obzir zajedno sa njihovim pozicijama unutar slike pri računanju konačnog deskriptora [221] [222] [223] [224]. Dodatno, ekstrahovanje deskriptora tekstura se jednostavno proširuje na slike u boji i multispektralne slike [225] [226]. Kako je ranije pomenuto, postprocesiranje, u ovom slučaju lokalnih obilježja može doprinijeti poboljšanju performansi pristupa baziranih na obilježjima niskog nivoa [227] [228].

S druge strane, pristupi bazirani na neuronskim mrežama, kako je detaljno obrazloženo, imaju tendenciju da daju daleko bolje rezultate zahvaljujući osobini neuronskih mreža (prvenstveno dubokih konvolucionih mreža) da uče lokalna obilježja višeg nivoa i uzimaju njhovu prostornu raspodjelu u obzir, kao posljedicu strukture mreže, bez potrebe za ručnom specifikacijom ove relacije. Uz navedene rezultate u oblasti detekcije objekata, konvolucione neuronske mreže pokazuju se kao izuzetno efikasne u klasifikaciji aero-snimaka, gdje najistaknutiji rezultati uključuju korištenje mreže radi pronalaženja hijerarhijskih struktura u slikama [229], korištenje različito skaliranih slika za poboljšavanje reprezentacije objekata koji variraju u veličini [230], kao i korištenje predobučavanih mreža za izdvajanje deskriptora obilježja i klasifikaciju aero-snimaka zemljišta visoke rezolucije [231]. Najznačajnije, pokazuje se, poređenjem značajnog broja postojećih pristupa, da se tačnost klasifikacije može poboljšati dodatnim obučavanjem mreže obučene na velikom skupu podataka [4].

Naivni pristupi koji koriste neuronske mreže i podrazumijevaju jednostavno obučavanje bez transfera ili razumijevanja prirode problema ne pokazuju značajno bolje rezultate od

postojećih pristupa baziranih na obilježjima niskog nivoa [232] [233], vrlo vjerovatno zbog nedostatka generalizacije i formiranja stvarnih obilježja visokog nivoa [217]. S druge strane, pokazuje se da se bolji rezultati mogu postići modifikacijom postojećih arhitektura [234], kako je i sugerisano deriviranom metodologijom u Glavi 5. Pored ovih pristupa, konstruisani su i dodatni, izuzetno specijalizovani, pristupi za rješavanje ovog problema, od kojih su najistaknutiji objektno-bazirana klasifikacija [235] koja modifikuje operaciju konvolucije radi izbjegavanja operacija na nivou piksela korištenjem objektnog segmentisanja ulaza, i hibridni MLP-CNN [236] klasifikator koji se bazira na pristupu sa ansamblom. Bitna karakteristika oba pomenuta pristupa je da, iako su vrlo usko specijalizovani i bazirani na specifičnom ciljanom istraživanju, i dalje vrlo blisko prate predloženi skup metoda za razvoj vještačkih neuronskih mreža, modelujući problem algoritamski, odnosno posmatrajući neuronsku mrežu kao algoritam sa učenim parametrima.

U periodu poslije publikacije predloženog pristupa i implementacije specijalizovane neuronske mreže za klasifikaciju i segmentaciju aero-snimaka [1], upotreba neuronskih mreža u ovoj oblasti se značajno povećala. Sa porastom interesovanja za ovu oblast, proširivani su i skupovi za obučavanje i testiranje, uključujući i prošireni NWPU-RESISC45 skup sa 45 klasa, koji predstavlja težu verziju istog problema koji je ovdje rješavan. Iako je ovaj prošireni skup konstruisan za upoređivanje performansi različitih pristupa, vrlo često specifični problemi zahtijevaju jedinstven skup, pa se veliki broj autora oslanja na analizu sopstvenih skupova ili razvoj novih skupova [237].

Pristup predložen u ovoj disertaciji, publikovan 2016. godine, pokazao se kao vrlo često korišten, pa se predobučavanje na poznatim modelima i fino podešavanje na konkretnom problemu pojavljuje i u novijim publikacijama [238] [239] [240], iako se za segmentaciju više preferiraju potpuno konvolucione neuronske mreže i konvolucioni autoenkoderi, zbog mogućnosti izvršavanja na novijem hardveru. Dodatno, savremeni pristupi obično uključuju neku kombinaciju neuronske mreže i algoritamskog pristupa, kako je i sugerisano predloženom metodologijom, pa se tako pojavljuje korištenje objektno-orijentisanih modela u kombinaciji sa konvolucionim neuronskim mrežama za klasifikaciju [235] [241] i korištenje kombinacija konvolucionih neuronskih mreža, analogno predloženoj modularnoj konstrukciji, za segmentaciju [242].

Savremeni pristupi za segmentaciju dominantno koriste varijante dubokih konvolucionih neuronskih mreža, primarno varijanti autoenkodera, kako bi postigle segmentaciju na nivou

piksela ulazne slike, bilo korištenjem preskačućih veza ili klasičnim konvolucionim enkoderima i dekoderima. Značajni primjeri ovakvog pristupa su korištenje dubokih konvolucionih mreža za mapiranje urbanih sredina [243] i Duboki U-Net [244] za segmentaciju zemljišta, baziran na U-Net [245] mreži za segmentaciju biomedicinskih slika.

6.3 MATERIJAL I METODOLOGIJA

6.3.1 Korišteni podaci

Za svrhu obuke i testiranja predložene mreže korištena su dva skupa podataka. Skup za obuku mreže bio je UCMerced skup slika aero-snimaka koji sadrži ručno označene slike iz USGS nacionalnih karata, sa 21 kategorijom i 100 slika po kategoriji, veličine 256×256 piksela (gdje jedan piksel predstavlja jednu stopu, odnosno 0.3m, na terenu). Ovaj skup se često koristio za evaluaciju efikasnosti klasifikacije, jer sadrži vizuelno slične, ali semantički različite kategorije. Slika 6.1 prikazuje kategorije iz ovog skupa.

Slika 6.1 – Primjeri različitih slika iz UCMerced skupa, redom slijeva udesno, odozgo ka dole: agrikulturalna oblast, avion, bejzbol teren, plaža, zgrade, čaparal, gusta rezidencijalna oblast, šuma, autoput, teren za golf, luka, raskrsnica, srednja rezidencijalna oblast, park za kamp-prikolice, nadvožnjak, parking, rijeka, pista, rijetka rezidencijalna oblast, kontejneri, teniski teren

Drugi skup podataka preuzet je direktno iz USGS nacionalne mape, u obliku nelabeliranih slika visoke rezolucije, veličine 5000×5000 piksela. Ovaj skup korišten je za validaciju tačnosti algoritma u slučaju problema segmentacije individualnih kategorija na većim kartama (odnosno, aero-snimcima). Pri validaciji tačnosti segmentacije, manuelno su evaluirane ćelije ulazne slike i upoređene sa labelama dobijenim iz mreže korištenjem opisanog algoritma. Ovaj skup je ilustrovan Slikom 6.2, gdje se vide primjeri različitih vrsta aero-snimaka koji se

potencijalno mogu koristiti za urbano planiranje, nadgledanje usjeva i slične primjene. Ovdje se takođe primjećuje i detekcija specifičnih vrsta objekata, kao što su teniski tereni, tereni za bejzbol, parkinzi i raskrsnice, čija bi manuelna identifikacija vizuelnom inspekcijom bila izuzetno spora i zahtjevna, s obzirom na sličnost susjednih objekata u urbanim sredinama. Za razliku od segmentacije ruralnih objekata i usjeva, koja se može izvesti i jednostavnijim pristupom, korištenjem obilježja niskog nivoa, segmentacija specijalizovanih objekata u urbanim sredinama može se smatrati zahtjevnijim zadatkom.

Slika 6.2 – Primjeri slika visoke rezolucije sa USGS karata. Sve slike su smanjene. Gore, lijevo: ruralna sredina sa izolovanim farmama. Gore, desno: naseljena urbana oblast. Dole, lijevo: srednje naseljena ruralna oblast. Dole, desno: luka i urbana oblast sa parkinzima i poljima.

Pored navedenog skupa, dodatni eksperimenti izvršeni su nad NWPU-RESISC45 skupom koji proširuje UCMerced skup na ukupno 45 klasa. Ovaj dodatni eksperiment opisan je u poglavlju 6.6.

6.3.2 Klasifikacija i segmentacija

Prvi izvedeni eksperiment uključuje klasifikaciju iskorištenog zemljišta na UCMerced skupu, sa podjelom skupa na skup za obučavanje i skup za testiranje korištenjem omjera 80:20. Mreža je obučena nad svim slikama iz skupa za obučavanje i validirana na slikama iz skupa za testiranje, kako bi rezultati bili uporedivi sa postojećim u literaturi. Pored toga, izvedeni su i testovi vremena izvršavanja algoritma segmentacije nad slikama visoke rezolucije, kao i vremena trajanja klasifikacije individualnih slika. Za ovu svrhu korišten je jedan GPGPU procesor, Nvidia GeForce 970 GTX i Intel i7 4790K procesor.

Drugi eksperiment imao je za cilj segmentaciju ulaznog aero-snimka visoke rezolucije na regione koji sadrže traženu klasu zemljišta. Prema pristupu izloženom u poglavlju 6.2, segmentacija je implementirana bez modifikacije arhitekture neuronske mreže, korištenjem klizećeg prozora, te je testirana na ručno labeliranom skupu, čiji je dio prikazan Slikom 6.3. Dodatno, pristup je konstruisan na takav način da može da se koristi za segmentaciju bilo koje klase koja je semantički slična klasama koje postoje u UCMerced skupu.

6.4 PREDLOŽENI PRISTUP ZA KLASIFIKACIJU I SEGMENTACIJU AERO-SNIMAKA

Na osnovu datog meta-algoritma, prvi korak je određivanje algoritamskog rješenja. Kako ulazni skup sadrži visok nivo šuma, ne postoji elegantno algoritamsko rješenje u literaturi, bez korištenja nekog vida obučavanja. Prema tome, potrebno je provjeriti postojanje nekog gotovog modela za rješavanje ovog problema. S obzirom na klasifikacionu prirodu problema i činjenicu da u trenutku projektovanja nije postojao gotov model za rješavanje ovog konkretnog problema, metodologija sugeriše selekciju i adaptaciju neke postojeće arhitekture. Kako bi se minimizirala potreba za obučavanjem mreže, prirodan izbor je između GoogleNet, ResNet i VGG mreža za koje postoje već obučeni modeli na ImageNet skupu.

Korišteni alati za rad sa neuronskim mrežama su Caffe [51] alat i Digits [199] platforma za obučavanje. Korištena je GoogleNet mreža, zbog dostupnosti za Caffe alat u trenutku konstrukcije, a parametri dobijeni obučavanjem nad ImageNet skupom su kopirani u novu mrežu, izuzev posljednjeg potpuno povezanog sloja za svaki od tri klasifikatora, čiji su parametri izmijenjeni u potpunosti, jer semantičko razumijevanje originalnih klasa na nivou

klasifikatora nije relevantno za dati problem. Potpuno povezani slojevi inicijalizovani su nasumično, Xavier inicijalizacijom. Slojevi nisu zaleđivani za vrijeme prve obuke, niti su skalirani koeficijenti učenja, te je mreža obučena do konvergencije. U hipotetskom slučaju da validacija nije dala dobre rezultate, tada bi se pristupilo zaleđivanju parametara ili skaliranju koeficijenta učenja na plićim slojevima. Razlog za čuvanje istog koeficijenta učenja nad postojećim slojevima je činjenica da ulazni skup aero-snimaka, vizuelnom inspekcijom, ne sadrži ista obilježja kao ImageNet skup, zbog načina akvizicije, ali je pretpostavka da su obilježja niskog nivoa relativno slična, pa se ne očekuje da greška na plićim slojevima bude značajna, dok se dozvoljava izmjena dubljih naučenih koncepata (odnosno obilježja višeg nivoa) za svrhu rješavanja problema klasifikacije aero-snimaka.

Radi povećavanja tačnosti mreže, upotrijebljen je pristup sa dvofaznim obučavanjem, prema prijedlozima iz poglavlja 5.4. Ukratko, mreža je obučena uz korištenje jednog optimizatora u prvoj fazi, a dodatno fino podešavana drugim optimizatorom u drugoj fazi. Ovaj pristup pokazao se kao efikasan za poboljšavanje konvergencije ka globalnom minimumu.

Za prvo obučavanje korišteno je klasično stohastičko gradijentsko spuštanje (SGD). Početna vrijednost koeficijenta učenja bila je 0.01, sa stepenastim smanjivanjem na 35% na svakih 700 iteracija (odnosno 10 epoha), a mreža je obučavana kroz 45 epoha (odnosno 3150 iteracija).

Druga faza obučavanja, odnosno drugo obučavanje, predstavljala je fino podešavanje mreže na istom skupu za obučavanje, korištenjem istog podskupa slika kao i u prvoj fazi. Ovdje je umjesto SGD pristupa, korišten adaptivni gradijent, gdje je $\gamma = 0.998$, sa eksponencijalno opadajućim koeficijentom učenja, počevši od 0.001, sa spuštanjem do 0.0002 na 20. epohi, gdje je obučavanje postiglo maksimalnu tačnost.

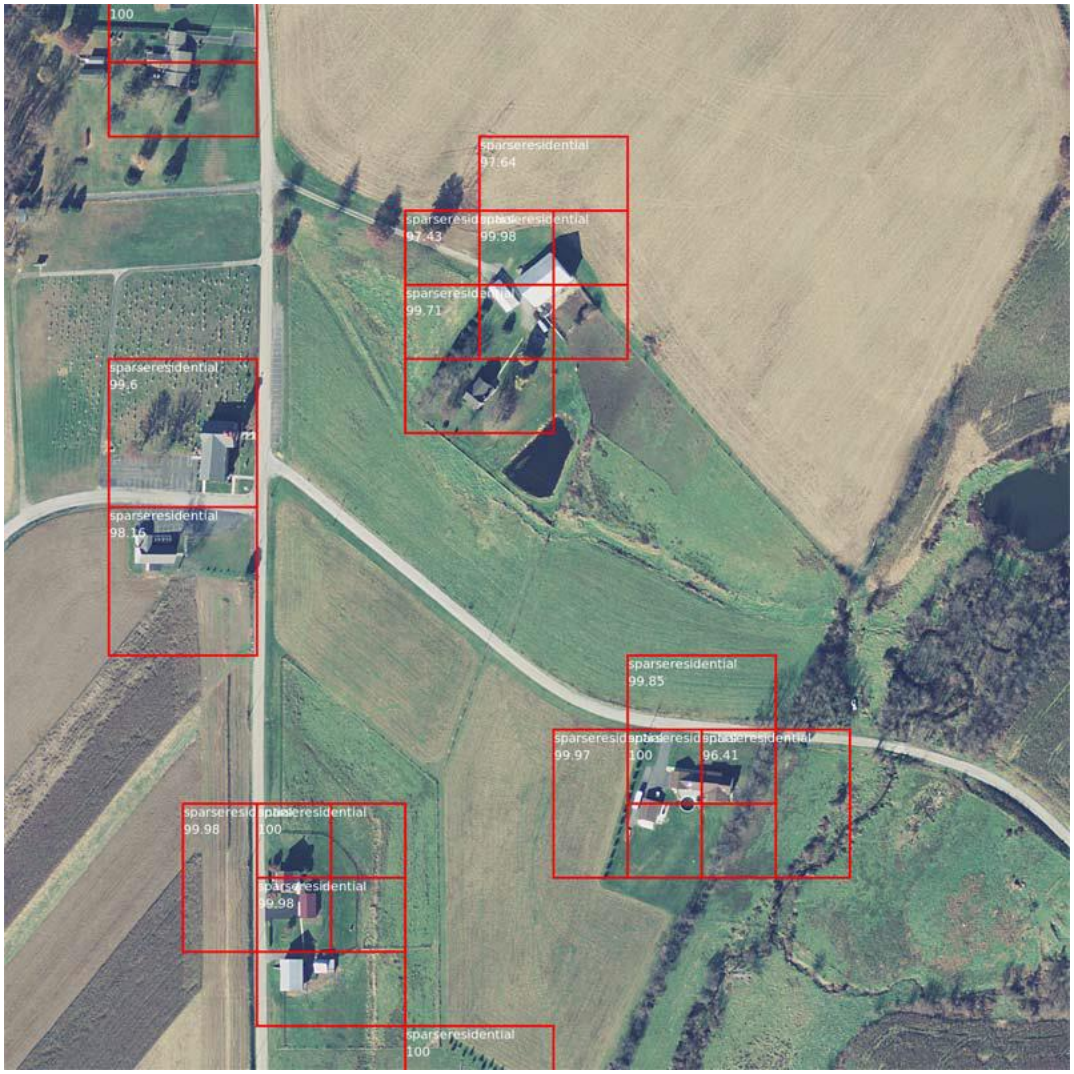
Prethodno objašnjeno rezonovanje za fino podešavanje je bilo glavna motivacija za ovakav pristup, pa su korišteni različiti algoritmi za obučavanje, pri čemu se fino podešavanje izvodilo sa početnom vrijednošću koeficijenta učenja većom od završne vrijednosti koeficijenta u prvoj fazi. Ovo omogućava mreži da se u drugoj fazi pomjeri iz potencijalnog lokalnog minimuma, odnosno da se model podesi ka boljem rješenju, izbjegavajući overfiting i povećavajući generalizaciju mreže. Pored inicijalnog skoka u vrijednosti koeficijenta učenja u drugoj fazi, koja omogućava pomjeranje iz lokalnog minimuma, dodatno fino podešavanje omogućava male promjene na fitovanoj funkciji sa ciljem bliže konvergencije u pronađeni minimum (odnosno smanjivanja fluktuacija oko minimuma).

Potreba za finim podešavanjem i povećavanjem generalizacije pojavila se usljed činjenice da bi se mreža morala koristiti za detekciju i segmentaciju zemljišta na slikama koje nisu korištene za vrijeme obučavanja. Sa visokom tačnošću mreže i visokim stepenom generalizacije, ovaj zadatak postaje izvodljiv.

Kako bi se minimiziralo vrijeme razvoja mreže, razvijeni klasifikacioni element algoritma rješenja, odnosno neuronska mreža za klasifikaciju, iskorišten je za svrhu segmentacije slike u ćelije zadate veličine korištenjem algoritamskog pristupa sa pomjerajućim prozorom, kako je sugerisano u Glavi 5. Ovaj pristup podrazumijevao je dijeljenje ulazne slike na preklapajuće segmente veličine 256×256 piksela, kako bi ulaz odgovarao ulazu očekivanom od Digits sistema koji je omogućio izvršavanje mreže na serverskoj strani i komunikaciju korištenjem REST (eng. *representation state transfer*) pristupa. Budući da je obučavanje rađeno na Digits sistemu, koji ujedno omogućava augmentaciju ulaza isijecanjem ulazne slike, ovaj sistem je očekivao ulaz iste veličine kako je definisano modelom (iako GoogleNet model prihvata ulaze veličine 224×224, Digits ulaze interno skalira). Za svaki od segmenata, vraćena je top-5 predikcija klase.

Korišten je horizontalni i vertikalni korak preklapanja od 128 piksela, gdje za svaki preklopljeni segment, veličine 128×128, postoje četiri top-5 rezultata, zbog preklapanja segmenata dimenzija 256×256. Za svaki preklopljeni segment izračunata je suma vrijednosti predikcija za svaku od potencijalnih klasa. Ukoliko klasa sa najvećom vjerovatnoćom prelazi predefinisani prag, ta ćelija se označava kao ćelija koja sadrži objekat detektovane klase. Primjer ovog pristupa dat je Slikom 6.3.

S ozbirom na predloženi meta-algoritam, ovo je ekvivalentno korištenju prethodno razvijene neuronske mreže za klasifikaciju kao „crne kutije“ u algoritmu koji prolazi kroz ulaznu sliku klizećim prozorom. Sa stanovišta rješavanja problema segmentacije, ono se svodi na drugi korak meta-algoritma, gdje se prethodno razvijena mreža za klasifikaciju koristi kao modul u algoritmu za segmentaciju. Dodatno, ovakav pristup omogućava analizu ulaznih slika proizvoljne veličine (ograničenu samo formatom koji se koristi i dostupnom radnom memorijom), te granularnost izlazne maske proizvoljne rezolucije. Suštinski, pristup je vrlo sličan R-CNN pristupu za detekciju, pri čemu se izlaz koristi kao maska sa proizvoljnom granularnošću, pa dobijeno rješenje može da se koristi i za detekciju objekata i za segmentaciju u klase.



Slika 6.3 – Dio Slike 6.4 na kom je izvedena detekcija klase nad ruralnom vrstom objekta (naznačene su i vjerovatnoće ispravnosti, kao i preklapajući segmenti)

Dakle, ovaj pristup omogućava da se algoritmu daje upit nad slikom proizvoljne rezolucije za segmentaciju i detekciju bilo koje klase za čije je prepoznavanje mreža obučena. Algoritam ne nameće ograničenja po pitanju veličine slike, pod uslovom da je slika dovoljne veličine da se može formirati jedan region od interesa (odnosno, slika mora biti minimalno one veličine koju zahtijeva arhitektura klasifikatora kako bi se izveo bar jedan korak klasifikacije).

6.5 REZULTATI

Rezultati prvog eksperimenta dati su Tabelom 6.1, gdje je dato poređenje postignutih rezultata sa najaktuelnijim u literaturi. Iz ove tabele se vidi da je, u slučaju pristupa korištenjem obilježja niskog nivoa, najbolji rezultat postignut u slučaju kada su obilježja drugog reda kombinovana u kompaktnu reprezentaciju nazvanu vektorima lokalno agregiranih tenzora

[228]. Ovdje je bitno naglasiti da dublja lokalna obilježja daju bolju diskriminativnu reprezentaciju tekstura. S obzirom na to da konvolucione neuronske mreže svojom arhitekturom omogućavaju slojevitu kombinaciju obilježja, te učenje obilježja izuzetno visokog nivoa, očekivano je da one daju bolje rezultate od pristupa sa obilježjima niskog nivoa. Pored rezultata metode sa dvofaznim obučavanjem, u Tabeli 6.1 je data i tačnost mreže poslije prve faze obučavanja (u tabeli predstavljena kao predložena stohastička metoda).

TABELA 6.1– POREĐENJE TAČNOSTI KLASIFIKACIJE AERO-SNIMAKA NA UCMERCED SKUPU PODATAKA, ZA PREDLOŽENI PRISTUP I POSTOJEĆE PRISTUPE U LITERATURI

Pristup baziran na obilježjima niskog nivoa	**Tačnost**
BoVW + SCK **[8]**	77.71%
SPCK+ **[221]**	76.05%
SPCK++ **[221]**	77.38%
BRSP **[222]**	77.80%
Nenadgledano učenje obilježja **[6]**	81.67%
mCENTRIST **[226]**	89.90%
VLAD **[228]**	92.50%
VLAT **[228]**	**94.30%**
PSR **[224]**	89.10%

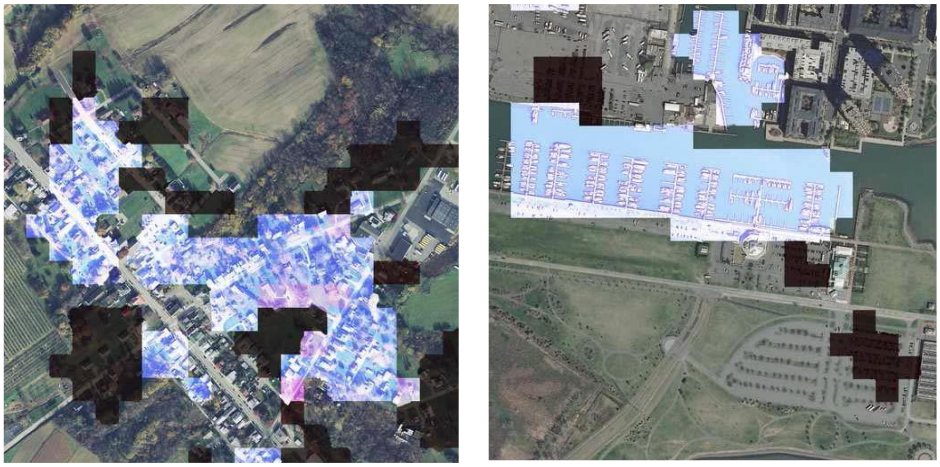
Pristup baziran na neuronskoj mreži	**Tačnost**
GoogleNet sa finim podešavanjem **[4]**	97.10%
ConvNet **[229]**	89.39%
IFK + VGG **[231]**	98.49%
Duboko učenje višestrukog pogleda **[230]**	93.48%
Predložena stohastička metoda	98.15%
Predložena alternativna jednofazna metoda	98.38%
Predložena dvofazna metoda	**98.61%**

Iz Tabele 6.1 se takođe vidi da većina nenaivnih pristupa koji koriste neuronske mreže daju bolje rezultate od svih pristupa baziranih na obilježjima niskog nivoa.

Pored pomenutih rezultata, nad istom mrežom je sproveden dodatni eksperiment, u Tabeli 6.1 označena kao predložena alternativna jednofazna metoda, gdje je mreža obučavana samo u prvoj fazi, pri čemu je obučavanje izvedeno u 100 epoha, sa početnom vrijednošću koeficijenta učenja 0.01 i stepenastim opadanjem na 50%, na svakih 16 epoha. Ovo podešavanje za jednu fazu obuke postiglo je maksimalnu tačnost od 98.38%. Izvođeni su dodatni eksperimenti sa jednofaznim obučavanjem, ali data konfiguracija dala je najbolje rezultate. Bitno je uočiti da dvofazno obučavanje i dalje daje bolje rezultate, konzistentno sa prezentovanom metodologijom.

Drugim eksperimentom testirana je efikasnost mreže pri detekciji objekata na aero-snimcima visoke rezolucije, te segmentaciji tih snimaka u regione od interesa. Slika 6.4 prikazuje rezultate segmentacije klasa na slikama visoke rezolucije. Ovdje se jasno vidi visoka tačnost segmentacije aero-snimaka zemljišta.

Formalno, proces segmentacije slike podrazumijeva da se svakoj ćeliji segmentiranog izlaza dodjeljuje klasa. Radi praktične primjenjivosti i optimizacije performansi, algoritam segmentiše sliku samo u one klase koje su od algoritma tražene upitom. Naravno, algoritam može da segmentiše sliku samo u one klase za koje je klasifikatorska mreža obučena, što je prirodno ponašanje bilo kog segmentacionog algoritma. Zbog toga, prikazana Slika 6.4 prikazuje ulazni snimak sa označenim segmentacijama za tražene klase, dok je ostatak ćelija segmentisan u klasu „ostalo“, jer njihove klase ili nisu specifikovane upitom algoritmu ili korišteni klasifikator ne sadrži odgovarajuću klasu za datu ćeliju.



Slika 6.4 – Rezultati automatske segmentacije na aero-snimcima visoke rezolucije. Gore, lijevo: rijetke rezidencijalne jedinice u ruralnom području. Gore, desno: označene zgrade i teren za bejsbol. Dole, lijevo: srednje naseljena rezidencijalna oblast i rijetko naseljena rezidencijalna oblast. Dole, desno: brodovi u luci i parkinzi

6.5.1 Performanse i skalabilnost

Pored toga što algoritam ne nameće ograničenja po pitanju veličine slike, potencijalno unapređenje performansi može da se izvede korištenjem većeg broja GPGPU jedinica na serverskoj strani ili distribuiranim izvršavanjem na većem broju mašina, tako da se omogući konkurentno klasifikovanje različitih ćelija mreže u koju je slika podijeljena. Za razliku od klasičnih modela za segmentaciju aero-snimaka (nastalih, hronološki, poslije razvoja ovog modela) koji se oslanjaju na potpuno konvolucione mreže (najčešće varijante konvolucionih autoenkodera), ovakav vid detekcije i segmentacije je visoko paralelizabilan i upotrebljiv u distribuiranom okruženju sa teoretski linearnim skaliranjem, prema Gustavsonovom zakonu [246], jer ne zahtijeva izmjene u arhitekturi mreže za ovaj zadatak, ne remeti tačnost rezultata pri distribuciji i ne unosi probleme sa sinhronizacijom.

Dodatno, kako se algoritam može parametrizovati, i kako postoji prostor za dodatna poboljšanja performansi korištenjem ažurnije verzije Digits sistema, ovaj algoritam može da se podesi i za druge zadatke klasifikacije aero-snimaka, jer je zasnovan na testiranoj metodologiji obuke mreže koja omogućava ponovljivost pri obučavanju, pa se nova mreža može sličnim pristupom obučiti za veći broj klasa, sa sitnijom podjelom slike i preciznijom segmentacijom.

Vrijeme inferencije predložene mreže za jednu sliku je 20ms, dok je vrijeme izvršavanja po slici visoke rezolucije 30.8 ± 0.8 sekundi. Ovdje je bitno ponovo naglasiti značajno ograničenje po pitanju načina prenosa slike na serversku stranu u izvedenom eksperimentu. S

obzirom na to da se radi o upisu na tvrdi disk i njegovom čitanju, ovo vrijeme bi moglo biti značajno kraće kada bi se prenos slika izvodio kroz mrežne kablove, zbog eliminacije potrebe za nasumičnim upisom i čitanjem na i sa tvrdog diska, te zbog nešto većeg propusnog opsega mrežnog interfejsa u odnosu na tvrdi disk. Dato vrijeme izvršavanja po slici visoke rezolucije uključivalo je procesiranje slike, lokalnu komunikaciju klijenta i serverske aplikacije i upis i učitavanje slike sa diska u GPU memoriju.

Posebno ograničenje koje je rezultat korištenja Digits alata je činjenica da je korištena verzija zahtijevala prosljeđivanje putanje do slike na fajl-sistemu, pa je za svaki izdvojeni segment mreže bilo neophodno upisivati sliku na disk i ponovo je učitavati u memoriju, što je vremenski zahtjevna operacija, budući da je korišten tvrdi disk rotacione brzine 7200 rotacija po minuti.

6.6 RJEŠENJE ZA VEĆI BROJ KLASA

Korištenjem ranije izloženog pristupa za klasifikaciju aero-snimaka i metodologije izložene u Glavi 5, adaptirana je GoogleNet mreža na isti način kako je izloženo u poglavlju 6.3, sa varijacijama navedenim u nastavku, i testirana na NWPU-RESISC45 skupu podataka. Ova studija sprovedena je kako bi se predloženi pristup validirao nad težim skupom podataka, sa ciljem pokazivanja dobre generalizacije i uporedivosti sa savremenim rezultatima u ovoj oblasti.

6.6.1 Parametrizacija i eksperimenti

S obzirom na to da je jedan od ciljeva ove studije bila validacija prethodno izloženog pristupa obučavanju mreže, implementiran je samo sistem za klasifikaciju, korištenjem istih verzija Caffe biblioteke i Digits alata za rad sa neuronskim mrežama.

Kao polazna mreža iskorištena je GoogleNet mreža predobučavana na ImageNet skupu, uz modifikaciju potpuno povezanih slojeva radi adaptacije na ciljnih 45 klasa. Ovi slojevi inicijalizovani su Xavier inicijalizacijom.

Mreža je obučavana u tri faze, analogno prethodno izloženoj verziji, pri čemu je dodata još jedna faza finog podešavanja.

U prvoj fazi, mreža je obučavana korištenjem Adam optimizatora sa početnom vrijednošću koeficijenta učenja 0.001, sa stepenastim opadanjem na 65%, na svakih 10 epoha, na ukupno 100 epoha. Veličina grupe bila je 512.

Druga faza koristila je fino podešavanje sa AdaDelta optimizatorom, sa eksponencijalno opadajućim koeficijentom učenja od 0.0005, sa koeficijentom opadanja 0.995, na 50 epoha. Veličina grupe bila je 512.

U posljednjoj, trećoj, fazi, obučavanje je rađeno ponovo sa Adam optimizatorom i eksponencijalno opadajućim koeficijentom učenja od 0.00005, sa koeficijentom opadanja 0.95, na 50 epoha. Veličina grupe bila je 1024.

Specifikovane veličine grupa uključuju akumulaciju gradijenata kroz više iteracija, zbog memorijskih ograničenja hardvera na kom je izvršavano obučavanje. Obučavanje je rađeno na dva GeForce 1080 Ti GPGPU procesora, korištenjem sistema za distribuciju posla integrisanog u Caffe alat. Ovdje je bitno napomenuti da, zbog ranije navedenih problema sa sinhronizacijom, distribuiranje posla na više GPGPU jedinica ovim alatom tipično daje nešto lošije rezultate (tipično unutar par procenata).

Validacija nad testnim skupom je rađena na svakih 10 epoha, radi bržeg obučavanja, budući da inferencija cijelog testnog skupa zahtijeva prolazak kroz 90% cijelog skupa.

Razlog za izmijenjene parametre optimizovanja ovdje je bio činjenica da je mreža obučavana na većem broju epoha, što je posljedica korištenja drugog skupa za obučavanje. Ovdje je NWPU-RESISC45 skup podijeljen u omjeru 10:90, radi validacije generalizacije ove mreže i poređenja sa rezultatima iz literature, gdje je, nad ovim skupom, uglavnom korišten ovaj omjer.

6.6.2 Rezultati klasifikacije

Nakon završetka obučavanja mreže u prvoj fazi, dobijena je tačnost od 85.39% na kraju obučavanja, pri čemu je maksimalna vrijednost u toku obučavanja bila 86.47%. Poslije prve faze finog podešavanja, odnosno poslije druge faze obučavanja, konačna dobijena tačnost mreže bila je 86.36%, a po završetku druge faze finog podešavanja, odnosno treće faze obučavanja, krajnja postignuta tačnost bila je 86.38%.

Ovi rezultati, kako je prikazano Tabelom 6.2, prevazilaze najbolje rezultate dobijene na ovom skupu korištenjem nadgledanog učenja.

TABELA 6.2– POREĐENJE TAČNOSTI KLASIFIKACIJE AERO-SNIMAKA NA NWPU-RESISC45 SKUPU PODATAKA, ZA PREDLOŽENI PRISTUP I POSTOJEĆE PRISTUPE [240] U LITERATURI

Korištena mreža	Tačnost
CaffeNet	77.62%
GoogleNet	76.00%
VGG-F	79.40%
VGG-S	80.00%
VGG-M	79.24%
VGG-16	78.64%
VGG-19	78.49%
ResNet	84.30%
Predloženo rješenje sa GoogleNet mrežom	**86.38%**
Nenadgledano učenje (najbolji rezultat)	88.60%

Ovdje je bitno uočiti da je korištena GoogleNet mreža koja je prethodno davala lošije rezultate u odnosu na druge arhitekture, što dodatno potvrđuje validnost predloženog pristupa obučavanju u više faza, kao i načina smanjivanja koeficijenta učenja.

Dodatno, ova mreža je obučena i validirana, na osnovu predložene metodologije unutar dvodnevnog perioda u kom je izvršeno i nekoliko dodatnih eksperimenata, radi potencijalnog pronalaženja boljeg rješenja.

Ovi dodatni eksperimenti su pokazali da korištenje dvograne mreže, kao u poglavlju 5.2.3, sa dvije GoogleNet podmreže, gdje je jedna predobučena na ImageNet skupu, a druga nasumično inicijalizovana, ne daje bolje rezultate od jedne predobučene mreže, sa dobijenom tačnošću klasifikacije od 74.46%, uz sličan pristup obučavanju, što je i očekivano, s obzirom na to da dodavanje nove mreže, bez fiksiranja predobučenih parametara, ovdje samo usložnjava arhitekturu i u suprotnosti je sa izloženom metodologijom.

Prema tome, izvedena proširena studija potvrđuje izloženu metodologiju i uspješno validira predloženi pristup implementacije specijalizovane neuronske mreže za klasifikaciju aero-snimaka. Takođe je validirana mogućnost proširivosti predloženog pristupa na dodatne klase i srodne probleme, testiranjem na proširenom i težem skupu podataka, uz značajno nepogodniju podjelu skupa za obučavanje. S obzirom da predloženi pristup segmentaciji

zemljišta može da se koristi sa bilo kojom mrežom za inferenciju, već postojeća implementacija sistema za segmentaciju može da se iskoristi nad proširenih 45 klasa.

Na osnovu ovoga može da se izvede zaključak da je implementirano rješenje za klasifikaciju i segmentaciju efikasno i primjenjivo u praksi, jer postiže vrhunske rezultate u oblasti klasifikacije i segmentacije aero-snimaka zemljišta, sa mogućnošću distribucije na više udaljenih mašina, kao i proširivanja na druge srodne zadatke i skupove podataka.

7 ZAKLJUČAK

Ova disertacija predstavila je i detaljno objasnila vlastitu implementaciju specijalizovane neuronske mreže za klasifikaciju i segmentaciju aero-snimaka zemljišta. Predstavljen je jedinstven način za projektovanje, adaptaciju i obučavanje duboke neuronske mreže za klasifikaciju aero-snimaka, te je dat algoritamski pristup za korištenje ove mreže za segmentaciju zemljišta u regione od interesa. Pored predstavljenog rješenja data je i metodologija neophodna da se razvije sistem slične namjene za rješavanje drugih klasifikacionih problema ili da se prezentovano rješenje adaptira za srodan problem. Predstavljeni su rezultati izvršavanja implementacije opisanog rješenja koji su pokazali izuzetno dobre rezultate, vrhunske u oblasti. Pored publikovanih rezultata, izvedeni su i dodatni eksperimenti nad proširenim skupom sa većim brojem klasa, gdje su takođe postignuti vrhunski rezultati, te je tako validiran i predstavljeni pristup i izložena metodologija na kojoj je on zasnovan.

Dat je detaljan pregled savremene oblasti mašinskog učenja, sa glavnim klasama arhitektura neuronskih mreža i načina njihove primjene, te je detaljno objašnjen matematički model neuronske mreže, uključujući CNN, FFNN i LSTM. Dato je vlastito izvođenje jednačina za propagaciju unazad i date su jednačine za sve savremene optimizatorske tehnike, aktivacione funkcije i funkcije gubitka. Na ovaj način je, na jednom mjestu, dat pregled ključnih elemenata neophodnih za razumijevanje izloženih apstraktnih principa projektovanja i adaptacije neuronskih mreža, uključujući i vlastito izvođenje jednačina propagacije unazad.

Prezentovani matematički model stavljen je u programski kontekst, gdje su objašnjene praktične metode konstrukcije neuronskih mreža na osnovu datog matematičkog modela. Uz programski model, dati su praktični savjeti i smjernice za konstrukciju arhitektura neuronskih mreža, izbor slojeva i parametara, izbjegavanje i prepoznavanje tipičnih problema, kao i poboljšavanje performansi mreža na različitim tipovima hardvera i platformi. Date preporuke i smjernice bazirani su na dugotrajnom i studioznom proučavanju cijele oblasti i sopstvenom iskustvu, uz odgovarajuće praktične primjere na sopstvenim i tuđim radovima. Ove smjernice predstavljaju svojevrsne putokaze za zaobilaženje čestih problema koji se mogu pojaviti pri razvoju sistema sa neuronskim mrežama, prvenstveno razvojnim inžinjerima iz oblasti izvan mašinskog učenja.

Pored programskog modela i praktičnih smjernica, dat je pregled najkorištenijih modela neuronskih mreža, za najčešće kategorije problema koje se javljaju u praksi, sa obrazloženjima njihove funkcionalnosti i isticanjem i objašnjenjem dominantnih osobina i specifičnosti tih modela sa ciljem praktične upotrebljivosti i u svrhu formiranja temelja za pristup razvoju ponuđenog rješenja za klasifikaciju i segmentaciju aero-snimaka. Iz sopstvenih eksperimenata konstruisani su savjeti o adaptacijama pomenutih mreža, njihovom dodatnom obučavanju, specijalizaciji ili algoritamskoj upotrebi u praktičnim, potencijalno komercijalnim, upotrebama.

Dodatno, na osnovu datog detaljnog pregleda, na osnovu postojeće literature i iz detaljnog vlastitog istraživanja i iskustva, prikazana je koherentna metodologija za projektovanje i razvoj specijalizovanih neuronskih mreža. Ova metodologija razvijana je u toku perioda istraživanja i na osnovu nje su konstruisane sve prezentovane i publikovane sopstvene arhitekture, te su arhitekturalne odluke iz prezentovanih, samostalno razvijenih, modela neuronskih mreža bile zasnovane na formiranoj metodologiji razvoja. Proces eksperimentisanja sa različitim arhitekturama, pristupima obučavanju i algoritamskoj augmentaciji, isprobavanja različitih parametara, formiranja rutiranja u mrežama i određivanja hiperparametara koji je konačno doveo do prezentovanih arhitektura pomogao je u formiranju predloženih metoda i pristupa za razvoj neuronskih mreža i datog meta-algoritma. Dugotrajni proces razvoja ovih metoda projektovanja i razvoja specijalizovanih neuronskih mreža, zasnovan na postojećoj literaturi i rezultatima drugih autora doveo je do konačnog proizvoda, specijalizovane neuronske mreže za klasifikaciju i segmentaciju aero-snimaka zemljišta, prezentovanog u ovoj disertaciji na primjerima vlastitih rezultata koji svjedoče o funkcionalnosti razvijenih metoda.

U trenutku objave ovog rada ne postoji jedna jasna metodologija za razvoj specijalizovanih neuronskih mreža, te je ovo jedan od mnogih iskoraka ka formiranju krajnje, objedinjujuće metodologije. Prezentovani pristup funkcionalan je u velikom broju praktičnih situacija za koje postoji akumulirano iskustvo i za koje je, na osnovu dosadašnjeg domenskog znanja, moguće formirati adekvatne preporuke i strategije. Ovaj skup metoda konstruisan je s ciljem olakšavanja i skraćenja vremena projektovanja i konstrukcije neuronskih mreža specijalizovanih za zadatke iz postojećih, dobro ustanovljenih, kategorija u nadi da će demistifikovati kompleksnu oblast mašinskog učenja i približiti je i staviti na raspolaganje široj publici inžinjera iz različitih oblasti, te da će napraviti još jedan iskorak ka formiranju sveobuhvatne metodologije projektovanja i razvoja neuronskih mreža.

Konačno i kao suštinski doprinos, razvijena metodologija primijenjena je u konkretnoj oblasti klasifikacije i segmentacije aero-snimaka zemljišta. Dobijeni rezultati pokazuju da dvofazno obučavanje neuronske mreže za svrhu klasifikacije daje izuzetno dobre rezultate, te da se mreža originalno konstruisana za klasifikaciju može efikasno adaptirati za svrhu segmentacije i detekcije, bez potrebe za dodatnim obučavanjem. Dodatno, dobijeni rezultati u toku istraživanja i izrade ove disertacije predstavljali su, u trenutku njihovog publikovanja, najbolje ostvarene u oblasti klasifikacije i segmentacije aero-snimaka, i predstavljaju demonstraciju funkcionalnosti dvofaznog pristupa obučavanju neuronskih mreža i adaptacije postojećih arhitektura neuronskih mreža za specijalizovane svrhe.

Prezentovano rješenje za klasifikaciju predstavljalo je jedno od prvih rješenja koje koristi učenje sa transferom i modifikaciju postojeće arhitekture neuronske mreže. Ono je takođe jedno od prvih koje ove pristupe koristi za klasifikaciju i segmentaciju aero-snimaka i jedno je od prvih koje implementira efikasnu segmentaciju aero-snimaka zemljišta korištenjem neuronskih mreža. Ova disertacija prikazuje cijeli proces razvoja ove arhitekture, sa svim arhitekturalnim odlukama donesenim radi poboljšavanja tačnosti klasifikacije i proširivanja skalabilnosti, efikasnosti i upotrebne vrijednosti segmentacionog pristupa. Prikazani su razlozi iz kojih je navedeni sistem bilo moguće projektovati, implementirati i testirati u dvosedmičnom periodu u vrijeme kada je ta oblast još uvijek bila u začetku.

Dodatno, analizirane su mogućnosti proširenja implementiranog sistema, uključujući distribuiranje na više GPGPU jedinica ili više mašina, povećavanje granularnosti izlaza segmentacije, korištenje sistema za detekciju objekata, te proširivanje sistema na dodatne klase za klasifikaciju, detekciju i segmentaciju.

Kako je ranije rečeno, implementirano rješenje obezbjeđuje platformu za dalje istraživanje, pogotovo u pogledu skalabilnosti implementiranog sistema za segmentaciju. Kako je rješenje implementirano kao višeslojna aplikacija čiji elementi komuniciraju preko mreže, a inferencija se izvodi u GPGPU okruženju, prirodan sljedeći korak je implementacija visoko distribuiranog sistema koji bi mogao da obrađuje velike količine podataka dobijenih udaljenom detekcijom.

Prezentovano rješenje i metodologija razvoja bazirani su isključivo na nadgledanom učenju i korištenju klasifikacionih modela za rad. Savremeni pristupi, koji daju najbolje rezultate u oblasti, baziraju se više na različitih varijantama izvođenja nenadgledanog ili polunadgledanog učenja, kako je prikazano u poglavlju 6.6. Ovo otvara nove puteve za

proširenje izložene uže metodologije razvoja, prvenstveno u smjeru enkapsulacije nenadgledanog učenja i sistematizacije pristupa koji koriste nenadgledano učenje. Nenadgledani i polunadgledani pristupi sve više se koriste u slučajevima kada je količina podataka neophodna za obučavanje nije dovoljna da bi se klasifikacioni ili segmentacioni zadatak riješio sa zadovoljavajućim nivoom tačnosti. Iako predloženo rješenje rješava veliki broj slučajeva klasifikacije i segmentacije, u vrlo specifičnim primjenama, kada je skup za obučavanje oskudan, pojavljuje se potreba za drugačijim pristupima, gdje se kao posebno obećavajuće ističe nenadgledano i polunadgledano obučavanje.

Kako tehnologija za akviziciju napreduje i kako se povećava propusni opseg za bežičnu komunikaciju sa bespilotnih letilica, otvaraju se mogućnosti obrade u realnom vremenu. Ovdje je, pored paralelizma sistema i skalabilnosti, potencijalno neophodno implementirati i vremenski zavisnu obradu, pogotovo u slučajevima kada se akvizicija radi sa letilica koje akviziciju rade relativno nisko.

Pored ovog načina obrade, alternativni način obrade aero-snimaka je korištenjem nekog ugrađenog sistema. Metodologija koja je ovdje izložena dijelom pokriva i ove slučajeve, jer su dati primjeri manje kompleksnih arhitektura mreža koje rješavaju slične probleme, ali kako se zahtjevi za obradom podataka povećavaju, tako se usložnjava i proces razvoja arhitektura specijalizovanih za ugrađene sisteme, pa se javlja potreba za proširivanjem metodologije razvoja i na ovu oblast.

Uprkos činjenici da je oblast primjene izuzetno široka, rješenje dato ovdje pokazuje izuzetnu primjenljivost i priširivost na druge domene, te otvara mogućnosti primjene u drugim oblastima i dalje istraživanje. Uz relevantnu studiju [1] čija je metodologija razvoja i razvojni proces ovdje opisan i čiji su rezultati prezentovani, ova disertacija prezentuje i proširenu studiju koja validira isti pristup nad proširenim i otežanim skupom podataka, NWPU-RESISC45, gdje su postignuti rezultati koji prevazilaze ostale dobijene nadgledanim učenjem, na istom proširenom skupu.

Pored implementiranog rješenja za klasifikaciju i segmentaciju koje postiže vrhunske rezultate, ova disertacija sadrži i svu neophodnu metodologiju za proširivanje i adaptaciju ovog istog sistema za druge skupove, nove klase i slične namjene, pa, prema tome, obezbjeđuje kompletno rješenje za klasifikaciju i segmentaciju aero-snimaka.

REFERENCE

[1] I. Ševo i A. Avramović, „Convolutional neural network based automatic object detection on aerial images,“ *IEEE geoscience and remote sensing letters,* t. 13, br. 5, str. 740-744, 2016.

[2] M. Omidalizarandi i M. Saadatseresht, „Segmentation and classification of point clouds from dense aerial image matching,“ *The International Journal of Multimedia & Its Applications,* t. 5, br. 4, 2013.

[3] V. Risojević, „Klasifikacija slika dobijenih tehnikama daljinske detekcije,“ Elektrotehnički fakultet Univerziteta u Banjoj Luci, Banja Luka, 2014.

[4] M. Castelluccio, G. Poggi, C. Sansone i L. Verdoliva, „Land use classification in remote sensing images by convolutional neural networks,“ arXiv.org, 2015.

[5] G. Cheng, J. Han i X. Lu, „Remote Sensing Image Scene Classification: Benchmark and State of the Art,“ u *Proceedings of the IEEE,* 2017.

[6] A. M. Cheriyadat, „Unsupervised Feature Learning for Aerial Scene Classification,“ *IEEE Transactions on Geoscience and Remote Sensing,* t. 52, br. 1, str. 439-451, 2014.

[7] USGS, „USGS Science Data Catalog,“ [Na mreži]. Link: https://data.usgs.gov/datacatalog/.

[8] Y. Yang i S. Newsam, „Bag-of-visual-words and spatial extensions for land-use classification,“ u *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems,* San Jose, 2010.

[9] Y. Lecun, L. Bottou, Y. Bengio i P. Haffner, „Gradient-based learning applied to document recognition,“ *Proceedings of the IEEE,* t. 86, br. 11, str. 2278-2324, 1998.

[10] AI Index, „Artificial Intelligence Index - 2018 Annual Report,“ 2018. [Na mreži]. Link: http://cdn.aiindex.org/2018/AI%20Index%202018%20Annual%20Report.pdf.

[11] Stanford University, Princeton University, „Image Net,“ Stanford University, Princeton University, 2018. [Na mreži]. Link: http://image-net.org/.

[12] K. He, X. Zhang, S. Ren i J. Sun, „Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,“ u *IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015.

[13] D. Silver, J. Schrittwieser, K. Simonyan i ostali, „Mastering the Game of Go without Human Knowledge,“ *Nature*, t. 550, br. 7676, str. 354-359, 2017.

[14] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau i S. Thrun, „Dermatologist-level classification of skin cancer with deep neural networks,“ *Nature*, t. 542, str. 115-118, 2017.

[15] Microsoft, „Microsoft reaches a historic milestone, using AI to match human performance in translating news from Chinese to English,“ Microsoft, 2018. [Na mreži]. Link: https://blogs.microsoft.com/ai/machine-translation-news-test-set-human-parity/.

[16] OpenAI, „OpenAI Five,“ OpenAI, 25 6 2018. [Na mreži]. Link: https://blog.openai.com/openai-five/.

[17] Google, „Improved Grading of Prostate Cancer Using Deep Learning,“ Google, 2018. [Na mreži]. Link: https://ai.googleblog.com/2018/11/improved-grading-of-prostate-cancer.html.

[18] I. Ševo i A. Avramović, „Multispectral scene recognition based on dual convolutional neural networks,“ u *10th International Symposium on Image and Signal Processing and Analysis*, Ljubljana, 2017.

[19] I. Ševo, „Semi-supervised Neural Network Training Method for Fast-moving Object Detection,“ u *NEUREL 2018*, Belgrade, 2018.

[20] A. Avramovic, R. Pilipovic, V. Stojnic, V. Jovanovic, I. Ševo, M. Simic, V. Risojević i Z. Babic, „Honeybee video-tracking for explosive detection,“ u *Mine Action 2018*, Slano, 2018.

[21] A. Avramovic, V. Jovanovic, R. Pilipovic, V. Stojnic, V. Risojević, S. Gajic, M. Simic, I. Ševo, M. Mustra, Z. Babic i J. Filipi, „Automatic monitoring of honeybees' activity outside of the hive from UHD video," u *NEUREL 2018*, Belgrade, 2018.

[22] I. Ševo i A. Kelečević, „Clustered class-dependant training method for digit recognition classifiers," u *INDEL 2016*, Banja Luka, 2016.

[23] I. Ševo i T. Mijatović, „Segmentation-Based Compound Figure Detection and Separation Methods," u *INDEL*, 2016.

[24] A. Avramović i I. Ševo, „Texture-based automatic polyp detection in colonoscopy videos," u *CMBEBIH 2015*, 2015.

[25] I. Ševo, „A framework for IoT sensor communication and prediction for small-scale systems," Elektrotehnički fakultet u Banjoj Luci, Banja Luka, 2017.

[26] I. Ševo, „Mehanizmi paralelizacije kompleksnih računarskih programa na desktop računarskim sistemima," Univerzitet u Banjoj Luci, Banja Luka; Završni rad II ciklusa studija, 2015.

[27] J. Vreeken, „Spiking neural networks, an introduction," *Physica A-statistical Mechanics and Its Applications*, 2003.

[28] Wikipedia, „Diagram of an artificial neuron," 2005. [Na mreži]. Link: https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Activation_Functions#/media/File:ArtificialNeuronModel_english.png.

[29] T. Isokawa, H. Nishimura i N. Matsui, „Quaternionic Multilayer Perceptron with Local Analyticity," *Information-an International Interdisciplinary Journal*, t. 3, br. 4, str. 756-770, 2012.

[30] D. C. Ciresan, U. Meier, L. M. Gambardella i J. Schmidhuber, „Deep, big, simple neural nets for handwritten digit recognition," *Neural Computation*, t. 22, br. 12, str. 3207-3220, 2010.

[31] R. P. Lippmann, „An introduction to computing with neural nets," *IEEE Assp Magazine*, t. 4, br. 2, str. 4-22, 1987.

[32] I. Goodfellow, Y. Bengio i A. Courville, Deep Learning (Adaptive Computation and Machine Learning series), The MIT Press, 2016.

[33] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang i S. Tandon, „Multi-Layer Neural Network," UFLDL Tutorial, [Na mreži]. Link: http://deeplearning.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/.

[34] J. McGonagle, G. Shaikouski i C. Williams, „Backpropagation," Brilliant.org, 2018. [Na mreži]. Link: https://brilliant.org/wiki/backpropagation/.

[35] A. Gibiansky, „Convolutional Neural Networks," 2014. [Na mreži]. Link: http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/.

[36] S. Russell i P. Norvig, Artificial Intelligence A Modern Approach, str. 578.

[37] J. V. Bouvrie, „Notes on Convolutional Neural Networks," , 2006. [Na mreži]. Link: http://cogprints.org/5869/1/cnn_tutorial.pdf. [Poslednji pristup 14 12 2018].

[38] K. Uchida, M. Tanaka, M. Tanaka i M. Okutomi, „Coupled convolution layer for convolutional neural network," *Neural Networks*, t. 105, br. , str. 197-205, 2018.

[39] V. Dumoulin i F. Visin, „A guide to convolution arithmetic for deep learning," 2018.

[40] H. Robbins i S. Monro, „A Stochastic Approximation Method," *Annals of Mathematical Statistics*, t. 22, br. 3, str. 400-407, 1951.

[41] J. Kiefer i J. Wolfowitz, „Stochastic Estimation of the Maximum of a Regression Function," *Annals of Mathematical Statistics*, t. 23, br. 3, str. 462-466, 1952.

[42] A. Karpathy, „Yes you should understand backprop," 19. 12. 2016. [Na mreži]. Link: https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b.

[43] S. Sharma, „Activation Functions: Neural Networks," 6. 9. 2017. [Na mreži]. Link: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[44] A. S. Walia, „Activation functions and it’s types - Which is better?,“ 29. 5. 2017. [Na mreži]. Link: https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f.

[45] A. Karpathy, „CS231n Convolutional Neural Networks for Visual Recognition,“ 28 11 2017. [Na mreži]. Link: https://github.com/cs231n/cs231n.github.io.

[46] B. Fortuner, „Activation Functions,“ 28 8 2018. [Na mreži]. Link: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html.

[47] X. Glorot, A. Bordes i Y. Bengio, „Deep Sparse Rectifier Neural Networks,“ u *Journal of Machine Learning Research* , 2010.

[48] R. Varma, „Picking Loss Functions - A comparison between MSE, Cross Entropy, and Hinge Loss,“ 9 1 2018. [Na mreži]. Link: http://rohanvarma.me/Loss-Functions/.

[49] M. Nielsen, „Improving the way neural networks learn,“ 10. 2018. [Na mreži]. Link: http://neuralnetworksanddeeplearning.com/chap3.html.

[50] O. Moindrot, „Triplet Loss and Online Triplet Mining in TensorFlow,“ 19. 3. 2018. [Na mreži]. Link: https://omoindrot.github.io/triplet-loss.

[51] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama i T. Darrell, „Caffe: Convolutional Architecture for Fast Feature Embedding,“ *arXiv: Computer Vision and Pattern Recognition*, t. , br. , str. 675-678, 2014.

[52] F. Chollet, „Keras Loss Layers,“ 2015. [Na mreži]. Link: https://keras.io/losses/.

[53] S. Ruder, „An overview of gradient descent optimization algorithms,“ 19. 1. 2016. [Na mreži]. Link: ruder.io/optimizing-gradient-descent.

[54] C. Darken, J. Chang i J. Moody, „Learning rate schedules for faster stochastic gradient search,“ u *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*, Helsingoer, Denmark, 1992.

[55] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli i Y. Bengio, „Identifying and attacking the saddle point problem in high-dimensional non-convex optimization,“ *arXiv: Learning,* str. 2933-2941, 2014.

[56] N. Qian, „On the momentum term in gradient descent learning algorithms,“ *Neural Networks,* t. 12, br. 1, str. 145-151, 1999.

[57] Y. E. Nesterov, „A method for solving the convex programming problem with convergence rate O (1/k^ 2),“ *Dokl. Akad. Nauk SSSR,* t. 269, str. 543-547, 1983.

[58] I. Sutskever, „Training Recurrent neural Networks,“ 2013.

[59] Y. Bengio, N. Boulanger-Lewandowski i R. Pascanu, „Advances in optimizing recurrent networks,“ *2013 IEEE International Conference on Acoustics, Speech and Signal Processing,* 2013.

[60] J. C. Duchi, E. Hazan i Y. Singer, „Adaptive Subgradient Methods for Online Learning and ...,“ *Journal of Machine Learning Research,* t. 12, 2011.

[61] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, K. Yang, Q. V. Le i A. Y. Ng, „Large Scale Distributed Deep Networks,“ , 2012. [Na mreži]. Link: https://ai.google/research/pubs/pub40565. [Poslednji pristup 3 1 2019].

[62] J. Pennington, R. Socher i C. D. Manning, „GloVe: Global Vectors for Word Representation,“ , 2014. [Na mreži]. Link: http://aclweb.org/anthology/d14-1162. [Poslednji pristup 3 1 2019].

[63] M. D. Zeiler, „ADADELTA: An Adaptive Learning Rate Method,“ *arXiv: Learning,* 2012.

[64] D. P. Kingma i J. Ba, „Adam: A Method for Stochastic Optimization,“ *arXiv: Learning,* 2015.

[65] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler i S. Hochreiter, „GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium,“ *arXiv: Learning,* str. 6626-6637, 2017.

[66] T. Dozat, „Incorporating Nesterov Momentum into Adam,“ u *2016*, ICLR 2016.

[67] G. Huang, Z. Liu, L. v. d. Maaten i K. Q. Weinberger, „Densely Connected Convolutional Networks,“ *arXiv: Computer Vision and Pattern Recognition,* t. , br. , str. 2261-2269, 2017.

[68] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes i J. Dean, „Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation,“ *Transactions of the Association for Computational Linguistics,* t. 5, br. 1, str. 339-351, 2016.

[69] S. J. Reddi, S. Kale i S. Kumar, „On the convergence of Adam and Beyond,“ u *Conference on Learning Representations*, 2018.

[70] EliteDataScience, „Overfitting in Machine Learning: What It Is and How to Prevent It,“ 7 9 2017. [Na mreži]. Link: https://elitedatascience.com/overfitting-in-machine-learning.

[71] P. Sharma, „Improving Neural Networks – Hyperparameter Tuning, Regularization, and More,“ 12 11 2018. [Na mreži]. Link: https://www.analyticsvidhya.com/blog/2018/11/neural-networks-hyperparameter-tuning-regularization-deeplearning/.

[72] T. Mandal, „Regularization: Hyperparameter tuning in a Neural Network,“ 29. 3. 2018. [Na mreži]. Link: https://medium.com/@tm2761/regularization-hyperparameter-tuning-in-a-neural-network-f77c18c36cd3.

[73] LO, „Differences between L1 and L2 as Loss Function and Regularization,“ 18. 12. 2013. [Na mreži]. Link: http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/.

[74] F. v. Veen, „The Neural Network Zoo,“ 14. 9. 2016. [Na mreži]. Link: http://www.asimovinstitute.org/neural-network-zoo/.

[75] F. Rosenblatt, „The perceptron: a probabilistic model for information storage and organization in the brain,“ *Psychological review,* str. 65-386, 1958.

[76] J. L. Elman, „Finding Structure in Time,“ *Cognitive Science,* t. 14, br. 2, str. 179–211, 1990.

[77] S. Hochreiter i J. Schmidhuber, „Long short-term memory,“ *Neural Computation,* t. 9, br. 8, 1997.

[78] C. Olah, „Understanding LSTM Networks,“ 27 8 2015. [Na mreži]. Link: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[79] Y. Bengio, P. Y. Simard i P. Frasconi, „Learning long-term dependencies with gradient descent is difficult,“ *EEE Transactions on Neural Networks,* t. 5, br. 2, str. 157-166, 1994.

[80] P. L. Carrier i K. Cho, „LSTM Networks for Sentiment Analysis,“ [Na mreži]. Link: http://deeplearning.net/tutorial/lstm.html.

[81] F. A. Gers, J. Schmidhuber i F. Cummins, „Learning to Forget: Continual Prediction with LSTM,“ *Neural Computation,* t. 12, br. 23, 2000.

[82] F. A. Gers i J. Schmidhuber, „Recurrent Nets that Time and Count,“ u *IEEE-INNS-ENNS International Joint Conference on Neural Networks,* Como, Italy, 2000.

[83] K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk i Y. Bengio, „Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,“ 2014.

[84] K. Yao, T. Cohn, K. Vylomova, K. Duh i C. Dyer, „Depth-Gated LSTM,“ 2015.

[85] J. Koutník, K. Greff, F. Gomez i J. Schmidhuber, „A Clockwork RNN,“ 2014.

[86] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink i J. Schmidhuber, „LSTM: A Search Space Odyssey,“ *IEEE Transactions on Neural Networks and Learning Systems,* t. 28, br. 10, str. 2222-2232, 2016.

[87] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Y. Bengio i Y. Bengio, „Empirical evaluation of gated recurrent neural networks on sequence modeling,“ *arXiv: Neural and Evolutionary Computing,* 2014.

[88] A. Graves, G. Wayne i I. Danihelka, „Neural Turing Machines," *arXiv: Neural and Evolutionary Computing,* 2014.

[89] H. Jaeger i H. Haas, „Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science,* t. 304, br. 5667, str. 78-80, 2004.

[90] J. J. Hopfield, „Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the National Academy of Sciences of the United States of America,* t. 79, br. 8, str. 2554-2558, 1982.

[91] S. Kirkpatrick, C. D. Gelatt i M. P. Vecchi, „Optimization by simulated annealing.," *Science,* t. 220, br. 4598, str. 671-680, 1983.

[92] G. E. Hinton i T. J. Sejnowski, „Learning and relearning in Boltzmann machines," , 1986. [Na mreži]. Link: https://dl.acm.org/citation.cfm?id=104291. [Poslednji pristup 2. 1. 2019].

[93] G. E. Hinton, „A Practical Guide to Training Restricted Boltzmann Machines," , 2012. [Na mreži]. Link: https://cs.toronto.edu/~hinton/absps/guidetr.pdf. [Poslednji pristup 2. 1. 2019].

[94] G. E. Hinton, „Training Products of Experts by Minimizing Contrastive ...," *Neural Computation,* t. 14, br. 8, str. 1771–1800, 2002.

[95] D. E. Rumelhart i J. L. McClelland, „Information Processing in Dynamical Systems: Foundations of Harmony Theory," u *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations,* MIT Press, 1987, str. 567.

[96] H. Bourlard i Y. Kamp, „Auto-association by multilayer perceptrons and singular value decomposition," *Biological Cybernetics,* t. 59, br. 4–5, str. 291–294, 1988.

[97] M. Ranzato, C. S. Poultney, S. Chopra i Y. L. Cun, „Efficient Learning of Sparse Representations with an Energy-Based Model," *Advances in Neural Information Processing Systems,* t. 19, 2006.

[98] D. P. Kingma i M. Welling, „Auto-Encoding Variational Bayes," 2013.

[99] P. Vincent, H. Larochelle, Y. Bengio i P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, 2008.

[100] Y. Bengio, P. Lamblin, D. Popovici i H. Larochelle, „Greedy Layer-Wise Training of Deep Networks,“ , 2006. [Na mreži]. Link: https://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf. [Poslednji pristup 2. 1. 2019.].

[101] M. D. Zeiler, D. Krishnan, G. W. Taylor i R. Fergus, „Deconvolutional networks,“ u *IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, 2010.

[102] T. D. Kulkarni, W. F. Whitney, P. Kohli i J. B. Tenenbaum, „Deep convolutional inverse graphics network,“ *arXiv: Computer Vision and Pattern Recognition*, str. 2539-2547, 2015.

[103] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville i Y. Bengio, „Generative Adversarial Nets,“ , 2014. [Na mreži]. Link: https://papers.nips.cc/paper/5423-generative-adversarial-nets. [Poslednji pristup 2. 1. 2019.].

[104] K. He, X. Zhang, S. Ren i J. Sun, „Deep Residual Learning for Image Recognition,“ 2015.

[105] G.-B. Huang, D. H. Wang i Y. Lan, „Extreme learning machines: a survey,“ *International Journal of Machine Learning and Cybernetics*, t. 2, br. 2, str. 107-122, 2011.

[106] W. Maass, T. Natschläger i H. Markram, „Real-time computing without stable states: a new framework for neural computation based on perturbations,“ *Neural Computation*, t. 14, br. 11, str. 2531-2560, 2002.

[107] D. Hebb, The organization of behavior, Wiley & Sons, 1949.

[108] A. Karpathy, „lossfunctions,“ [Na mreži]. Link: https://lossfunctions.tumblr.com/.

[109] R. Collobert i J. Weston, „A unified architecture for natural language processing: deep neural networks with multitask learning,“ u *25th international conference on Machine learning*, New York, 2008.

[110] L. Deng, G. Hinton i B. Kingsbury, „New types of deep neural network learning for speech recognition and related applications: an overview," u *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, 2013.

[111] R. B. Girshick, „Fast R-CNN," *arXiv: Computer Vision and Pattern Recognition*, str. 1440-1448, 2015.

[112] B. Ramsundar, S. Kearnes, P. F. Riley, D. R. Webster, D. E. Konerding i V. S. Pande, „Massively Multitask Networks for Drug Discovery," *arXiv: Machine Learning*, 2015.

[113] S. Ruder, „An Overview of Multi-Task Learning in Deep Neural Networks," 29 5 2017. [Na mreži]. Link: ruder.io/multi-task.

[114] R. Caruana, „Multitask learning: a knowledge-based source of inductive bias," , 1993. [Na mreži]. Link: http://dblp.uni-trier.de/db/conf/icml/icml1993.html. [Poslednji pristup 4. 1. 2019.].

[115] J. Baxter, „A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling," *Machine Learning*, t. 28, br. 1, str. 7-39, 1997.

[116] L. Duong, T. Cohn, S. Bird i P. Cook, „Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser," u *7th International Joint Conference on Natural Language Processing*, 2015.

[117] Y. Yang i T. M. Hospedales, „Trace Norm Regularised Deep Multi-Task Learning.," *arXiv: Learning*, 2016.

[118] Y. S. Abu-Mostafa, „Learning from hints in neural networks," *Journal of Complexity*, t. 6, br. 2, str. 192-198, 1992.

[119] J. Baxter, „A Model of Inductive Bias Learning," *Journal Of Artificial Intelligence Research*, t. 12, str. 149-198, 2011.

[120] K. Zhai i H. Wang, „Adaptive Dropout with Rademacher Complexity Regularization," u *International Conference on Learning Robots*, Vienna, 2018.

[121] R. Pascanu, T. Mikolov i Y. Bengio, „On the difficulty of training Recurrent Neural Networks,“ arXiv.org, 2013.

[122] J. Brownlee, „A Gentle Introduction to Exploding Gradients in Neural Networks,“ 18. 12. 2017. [Na mreži]. Link: https://machinelearningmastery.com/exploding-gradients-in-neural-networks/.

[123] E. Alese, „The curious case of the vanishing & exploding gradient,“ 5 6 2018. [Na mreži]. Link: https://medium.com/learn-love-ai/the-curious-case-of-the-vanishing-exploding-gradient-bf58ec6822eb.

[124] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu i X. Zheng, „TensorFlow: a system for large-scale machine learning,“ *arXiv: Distributed, Parallel, and Cluster Computing*, t. , br. , str. 265-283, 2016.

[125] Y. Bengio, J. Louradour, R. Collobert i J. Weston, „Curriculum learning,“ u *26th Annual International Conference on Machine Learning*, New York, 2009.

[126] W. Zaremba i I. Sutskever, „Learning to Execute,“ arXiv.org, 2015.

[127] S. Ioffe i C. Szegedy, „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,“ *arXiv: Learning*, str. 448-456, 2015.

[128] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach i J. Martens, „Adding Gradient Noise Improves Learning for Very Deep Networks,“ arXiv.org, 2015.

[129] Microsoft, „Layers Library Reference,“ 2017. [Na mreži]. Link: https://www.cntk.ai/pythondocs/layerref.html.

[130] Google, „Module: tf.layers,“ 20 11 2018. [Na mreži]. Link: https://www.tensorflow.org/api_docs/python/tf/layers.

[131] V. Dumoulin i F. Visin, „A guide to convolution arithmetic for deep learning,“ arXiv.org, 2018.

[132] M. Lin, Q. Chen i S. Yan, „Network In Network,“ *arXiv: Neural and Evolutionary Computing,* 2014.

[133] M. D. Zeiler i R. Fergus, Visualizing and Understanding Convolutional Networks, New York: Courant Institute, New York University, 2013.

[134] M. Long, Z. Cao, J. Wang i P. S. Yu, „Learning Multiple Tasks with Deep Relationship Networks,“ arXiv.org, 2017.

[135] I. Misra, A. Shrivastava, A. Gupta i M. Hebert, „Cross-Stitch Networks for Multi-task Learning,“ *arXiv: Computer Vision and Pattern Recognition,* str. 3994-4003, 2016.

[136] M. Tan, „MnasNet: Towards Automating the Design of Mobile Machine Learning Models,“ 7. 8. 2018. [Na mreži]. Link: https://ai.googleblog.com/2018/08/mnasnet-towards-automating-design-of.html.

[137] J. Appleyard i S. Yokim, „Programming Tensor Cores in CUDA 9,“ 17 10 2017. [Na mreži]. Link: https://devblogs.nvidia.com/programming-tensor-cores-cuda-9/.

[138] L. Brown, „cuDNN v2: Higher Performance for Deep Learning on GPUs,“ 31 3 2015. [Na mreži]. Link: https://devblogs.nvidia.com/cudnn-v2-higher-performance-deep-learning-gpus/.

[139] L. Brown, „Accelerate Machine Learning with the cuDNN Deep Neural Network Library,“ 7. 9. 2014. [Na mreži]. Link: https://devblogs.nvidia.com/accelerate-machine-learning-cudnn-deep-neural-network-library/.

[140] F. Niu, B. Recht, C. Ré i S. J. Wright, „HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent,“ arXiv.org, 2011.

[141] H. B. Mcmahan, Google, S. Wa i M. Streeter, „Delay-Tolerant Algorithms for Asynchronous Distributed Online Learning,“ u *Advances in Neural Information Processing Systems*, 2014.

[142] S. Zhang, A. Choromanska i Y. LeCun, „Deep learning with elastic averaging SGD,“ *arXiv: Learning,* str. 685-693, 2015.

[143] Microsoft, Amazon, Facebook, „Open Neural Network Exchange Format,“ 4. 12. 2018. [Na mreži]. Link: https://onnx.ai/.

[144] R. Venkatesan i B. Li, „Diving deeper into mentee networks.,“ *arXiv: Learning,* 2016.

[145] S. Biook, „Migrate Deep Learning Training onto Mobile Devices!,“ 24. 4. 2017. [Na mreži]. Link: https://becominghuman.ai/part-1-migrate-deep-learning-training-onto-mobile-devices-c28029ffeb30.

[146] I. Goodfellow, Y. Bengio i A. Courville, Deep Learning, MIT Press, 2016.

[147] J. Brownlee, „Why Initialize a Neural Network with Random Weights?,“ 1. 8. 2018. [Na mreži]. Link: https://machinelearningmastery.com/why-initialize-a-neural-network-with-random-weights/.

[148] X. Glorot i Y. Bengio, „Understanding the difficulty of training deep feedforward neural networks,“ *Journal of Machine Learning Research,* t. 9, str. 249-256, 2010.

[149] A. Peruničić, „Choosing Weights: Small Changes, Big Differences,“ 25. 7. 2017. [Na mreži]. Link: https://intoli.com/blog/neural-network-initialization/.

[150] A. Saeed, „Using Genetic Algorithm for optimizing Recurrent Neural Network,“ 18. 8. 2018. [Na mreži]. Link: http://aqibsaeed.github.io/2017-08-11-genetic-algorithm-for-optimizing-rnn/.

[151] D. Ulyanov, A. Vedaldi i V. Lempitsky, „Instance Normalization: The Missing Ingredient for Fast Stylization,“ arXiv.org, 2017.

[152] M. Mozifian, „Normalization Techniques - Neural Networks,“ 11. 8. 2018. [Na mreži]. Link: https://melfm.github.io/posts/2018-08-Understanding-Normalization/.

[153] A. Krizhevsky, I. Sutskever i G. Hinton, „ImageNet Classification with Deep Convolutional Neural Networks,“ *NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada*

[154] S.-H. Tsang, „Review: AlexNet, CaffeNet — Winner of ILSVRC 2012 (Image Classification),“ 9. 8. 2019. [Na mreži]. Link: https://medium.com/coinmonks/paper-

review-of-alexnet-caffenet-winner-in-ilsvrc-2012-image-classification-b93598314160.

[155] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg i L. Fei-Fei, „ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, t. 115, br. 3, str. 211-252, 2015.

[156] K. Simonyan i A. Zisserman, „Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv: Computer Vision and Pattern Recognition*, 2015.

[157] koustubh, „ResNet, AlexNet, VGGNet, Inception: Understanding various architectures of Convolutional Networks," 3. 8. 2018. [Na mreži]. Link: https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/.

[158] A. Rosebrock, „ImageNet: VGGNet, ResNet, Inception, and Xception with Keras," 20. 3. 2017. [Na mreži]. Link: https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/.

[159] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke i A. Rabinovich, „Going deeper with convolutions," *arXiv: Computer Vision and Pattern Recognition*, str. 1-9, 2015.

[160] V. Chu, „We Need to Go Deeper: A Practical Guide to Tensorflow and Inception," 13. 2. 2017. [Na mreži]. Link: https://medium.com/initialized-capital/we-need-to-go-deeper-a-practical-guide-to-tensorflow-and-inception-50e66281804f.

[161] K. He, X. Zhang, S. Ren i J. Sun, „Identity Mappings in Deep Residual Networks," *arXiv: Computer Vision and Pattern Recognition*, str. 630-645, 2016.

[162] C. Szegedy, S. Ioffe, V. Vanhoucke i A. A. Alemi, „Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *arXiv: Computer Vision and Pattern Recognition*, str. 4278-4284, 2016.

[163] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens i Z. Wojna, „Rethinking the Inception Architecture for Computer Vision,“ *arXiv: Computer Vision and Pattern Recognition*, str. 2818-2826, 2016.

[164] F. Chollet, „Xception: Deep Learning with Depthwise Separable Convolutions,“ *arXiv: Computer Vision and Pattern Recognition*, str. 1800-1807, 2017.

[165] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally i K. Keutzer, „SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size,“ arXiv.org, 2016.

[166] R. B. Girshick, J. Donahue, T. Darrell i J. Malik, „Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,“ *arXiv: Computer Vision and Pattern Recognition*, str. 580-587, 2014.

[167] S. Ren, K. He, R. B. Girshick i J. Sun, „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, t. 39, br. 6, str. 1137-1149, 2017.

[168] R. Gandhi, „R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms,“ 9. 7. 2018. [Na mreži]. Link: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

[169] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu i A. C. Berg, „SSD: Single Shot MultiBox Detector,“ *arXiv: Computer Vision and Pattern Recognition*, str. 21-37, 2016.

[170] J. Hui, „SSD object detection: Single Shot MultiBox Detector for real-time processing,“ 14. 5. 2018. [Na mreži]. Link: https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06.

[171] E. Forson, „Understanding SSD MultiBox—Real-Time Object Detection In Deep Learning,“ 18. 11. 2017. [Na mreži]. Link: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab.

[172] C. Szegedy, S. E. Reed, D. Erhan i D. Anguelov, „Scalable, high-quality object detection,“ *arXiv: Computer Vision and Pattern Recognition,* 2015.

[173] J. Redmon, S. Divvala, R. Girshick i A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection,“ arXiv.org, 2016.

[174] J. Redmon i A. Farhadi, „YOLO9000: Better, Faster, Stronger,“ arXiv.org, 2016.

[175] M. Chablani, „YOLO - You only look once, real time object detection explained,“ 21 8 2017. [Na mreži]. Link: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006.

[176] J. Le, „How to do Semantic Segmentation using Deep learning,“ 3 5 2018. [Na mreži]. Link: https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef.

[177] E. Shelhamer, J. Long i T. Darrell, „Fully Convolutional Networks for Semantic Segmentation,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence,* t. 39, br. 4, str. 640-651, 2017.

[178] V. Badrinarayanan, A. Kendall i R. Cipolla, „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, t. 39, br. 12, str. 2481-2495, 2017.

[179] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy i A. L. Yuille, „Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs,“ arXiv.org, 2016.

[180] F. Yu i V. Koltun, „Multi-Scale Context Aggregation by Dilated Convolutions,“ *arXiv: Computer Vision and Pattern Recognition,* 2016.

[181] L. A. d. Santos, „Image Segmentation,“ [Na mreži]. Link: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html.

[182] H. Noh, S. Hong i B. Han, „Learning Deconvolution Network for Semantic Segmentation,“ *arXiv: Computer Vision and Pattern Recognition,* str. 1520-1528, 2015.

[183] P. Dar, „Top 10 Pretrained Models to get you Started with Deep Learning (Part 1 - Computer Vision),“ 27. 7. 2018. [Na mreži]. Link: https://www.analyticsvidhya.com/blog/2018/07/top-10-pretrained-models-get-started-deep-learning-part-1-computer-vision/.

[184] K. He, G. Gkioxari, P. Dollár i R. Girshick, „Mask R-CNN,“ arXiv.org, 2018.

[185] A. Radford, L. Metz i S. Chintala, „Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,“ *arXiv: Learning,* 2016.

[186] OpenAI, „Generative Models,“ 16. 6. 2016. [Na mreži]. Link: https://blog.openai.com/generative-models/.

[187] skymind.ai, „A Beginner's Guide to Generative Adversarial Networks (GANs),“ [Na mreži]. Link: https://skymind.ai/wiki/generative-adversarial-network-gan.

[188] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford i X. Chen, „Improved Techniques for Training GANs,“ arXiv.org, 2016.

[189] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever i P. Abbeel, „InfoGAN: interpretable representation learning by information maximizing generative adversarial nets,“ *arXiv: Learning,* str. 2180-2188, 2016.

[190] A. Brock, J. Donahue i K. Simonyan, „Large Scale GAN Training for High Fidelity Natural Image Synthesis.,“ *arXiv: Learning,* 2018.

[191] T. Karras, S. Laine i T. Aila, „A Style-Based Generator Architecture for Generative Adversarial Networks,“ arXiv.org, 2018.

[192] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung i C. Schroers, „A Fully Progressive Approach to Single-Image Super-Resolution,“ arXiv.org, 2018.

[193] L. A. Gatys, A. S. Ecker i M. Bethge, „A Neural Algorithm of Artistic Style,“ arXiv.org, 2015.

[194] R. Yuan, „Neural Style Transfer: Creating Art with Deep Learning using tf.keras and eager execution,“ 3. 8. 2018. [Na mreži]. Link: https://medium.com/tensorflow/neural-style-transfer-creating-art-with-deep-learning-using-tf-keras-and-eager-execution-7d541ac31398.

[195] C. Gao, D. Gu, F. Zhang i Y. Yu, „ReCoNet: Real-time Coherent Video Style Transfer Network.,“ *arXiv: Computer Vision and Pattern Recognition*, 2018.

[196] NeuroHive, „Real-time Video Style Transfer: Fast, Accurate and Temporally Consistent,“ 19 7 2018. [Na mreži]. Link: https://medium.com/coinmonks/real-time-video-style-transfer-fast-accurate-and-temporally-consistent-863a175e06dc.

[197] I. Sevo, A. Avramovic, I. Balasingham, O. J. Elle, J. Bergsland i L. Aabakken, „Edge density based automatic detection of inflammation in colonoscopy videos,“ *Computers in Biology and Medicine*, t. 72, str. 138-150, 2016.

[198] Microsoft, „Microsoft Cognitive Toolkit (CNTK), an open source deep-learning toolkit,“ 9. 1. 2019. [Na mreži]. Link: https://github.com/Microsoft/CNTK.

[199] Nvidia, „Deep Learning DIGITS,“ 19. 12. 2018. [Na mreži]. Link: https://docs.nvidia.com/deeplearning/digits/index.html.

[200] N. Salamati, D. Larlus i G. Csurka, „Combining visible and near-infrared cues for image categorisation,“ u *British Machine Vision Conference*, 2011.

[201] I. Ševo, „Stabilizacija generativnih suparničkih mreža na primjeru rukom pisanih cifara,“ Elektrotehnički fakultet u Banjoj Luci, Banja Luka; Izvještaj sa studije, 2018.

[202] T. Karras, T. Aila, S. Laine i J. Lehtinen, „Progressive Growing of GANs for Improved Quality, Stability, and Variation,“ arXiv.org, 2018.

[203] Y. LeCun, C. Cortes i C. J. Burges, „The MNIST database of handwritten digits,“ 1998. [Na mreži]. Link: http://yann.lecun.com/exdb/mnist/.

[204] M. Mohammadi i S. Das, „SNN: Stacked Neural Networks,“ arXiv.org, 2016.

[205] J. Yosinski, J. Clune, Y. Bengio i H. Lipson, „How transferable are features in deep neural networks?,“ arXiv.org, 2014.

[206] A. S. Razavian, H. Azizpour, J. Sullivan i S. Carlsson, „CNN Features Off-the-Shelf: An Astounding Baseline for Recognition,“ u *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Columbus, OH, 2014.

[207] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki i S. Carlsson, „From generic to specific deep representations for visual recognition,“ u *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Boston, MA, 2015.

[208] S. Xie, A. Kirillov, R. Girshick i K. He, „Exploring Randomly Wired Neural Networks for Image Recognition,“ arXiv.org, 2019.

[209] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le i A. Kurakin, „Large-Scale Evolution of Image Classifiers,“ u *34th International Conference on Machine Learning*, Sydney, 2017.

[210] K. O. Stanley i R. Miikkulainen, „Evolving Neural Networks through Augmenting Topologies,“ *Evolutionary Computation*, t. 10, br. 2, str. 99-127, 2002.

[211] E. Real, A. Aggarwal, Y. Huang i Q. V. Le, „Regularized Evolution for Image Classifier Architecture Search,“ arXiv.org, 2018.

[212] B. Zoph i Q. V. Le, „Neural Architecture Search with Reinforcement Learning,“ u *International Conference on Learning Representations*, 2017.

[213] A. Gotmare, N. S. Keskar, C. Xiong i R. Socher, „A Closer Look at Deep Learning Heuristics: Learning rate restarts, Warmup and Distillation,“ arXiv.org, 2018.

[214] I. Ševo, „Fast approximate subcutaneous tissue simulation algorithm for real-time rendering applications,“ Elektrotehnički fakultet u Banjoj Luci, Banja Luka; Izvještaj sa studije, 2018.

[215] I. Ševo, S. Lekić i M. Savić, „Self-Avoiding Hamiltonian Walks Counting in Parallel Processing Mode,“ u *High-Performance Computing Infrastructure for South East Europe's Research Communities*, Springer, 2012, str. 59-66.

[216] T. B. Lesson, „The Bitter Lesson,“ 13. 3. 2019. [Na mreži]. Link: http://www.incompleteideas.net/IncIdeas/BitterLesson.html.

[217] W. Brendel i M. Bethge, „Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet,“ u *International Conference on Learning Representations*, 2019.

[218] D. G. Lowe, „Distinctive Image Features from Scale-Invariant Keypoints,“ *International Journal of Computer Vision*, t. 60, br. 2, str. 91-110, 2004.

[219] N. Dalal i B. Triggs, „Histograms of oriented gradients for human detection,“ u *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, 2005.

[220] M. Shahriari i R. Bergevin, „Land-use scene classification: a comparative study on bag of visual word framework,“ *Multimedia Tools and Applications*, t. 76, br. 21, str. 23059-23075, 2017.

[221] Y. Yang i S. Newsam, „Spatial pyramid co-occurrence for image classification,“ u *2011 International Conference on Computer Vision*, Barcelona, 2011.

[222] Y. Jiang, J. Yuan i G. Yu, „Randomized Spatial Partition for Scene Recognition,“ u *European Conference on Computer Vision*, 2012.

[223] W.-L. Zhao, G. Gravier i H. Jegou, „Oriented pooling for dense and non-dense rotation-invariant features,“ u *British Machine Vision Conference 2013*, 2013.

[224] S. Chen i Y. Tian, „Pyramid of Spatial Relatons for Scene-Level Land Use Classification,“ *IEEE Transactions on Geoscience and Remote Sensing*, t. 53, br. 4, str. 1947-1957, 2015.

[225] M. Brown i S. Süsstrunk, „Multi-spectral SIFT for scene category recognition,“ u *CVPR 2011*, Colorado Springs, 2011.

[226] Y. Xiao, J. Wu i J. Yuan, „mCENTRIST: A Multi-Channel Feature Generation Mechanism for Scene Categorization,“ *IEEE Transactions on Image Processing* , t. 23, br. 2, str. 823-836, 2014.

[227] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez i C. Schmid, „Aggregating Local Image Descriptors into Compact Codes,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence,* t. 34, br. 9, str. 1704-1716, 2012.

[228] R. Negrel, D. Picard i P.-H. Gosselin, „Evaluation of second-order visual features for land-use classification,“ u *2014 12th International Workshop on Content-Based Multimedia Indexing (CBMI)*, Klagenfurt, 2014.

[229] K. Nogueira, W. O. Miranda i J. A. D. Santos, „Improving Spatial Feature Representation from Aerial Scenes by Using Convolutional Networks,“ u *2015 28th SIBGRAPI Conference on Graphics, Patterns and Images*, Salvador, 2015.

[230] F. P. S. Luus, B. P. Salmon, F. v. d. Bergh i B. T. J. Maharaj, „Multiview Deep Learning for Land-Use Classification,“ *IEEE Geoscience and Remote Sensing Letters* , t. 12, br. 12, str. 2448-2452, 2015.

[231] F. Hu, G.-S. Xia, J. Hu i L. Zhang, „Transferring Deep Convolutional Neural Networks for the Scene Classification of High-Resolution Remote Sensing Imagery,“ *Remote Sensing,* t. 7, br. 11, str. 14680-14707, 2015.

[232] Y. Zhu i S. Newsam, „Land use classification using convolutional neural networks applied to ground-level images,“ u *23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems* , Seattle, 2015.

[233] Y. Hu, Q. Zhang, Y. Zhang i H. Yan, „A Deep Convolution Neural Network Method for Land Cover Mapping: A Case Study of Qinhuangdao, China,“ *Remote Sensing,* t. 10, br. 12, str. 2053, 2018.

[234] G. J. Scott, M. R. England, W. A. Starms, R. A. Marcum i C. H. Davis, „Training Deep Convolutional Neural Networks for Land–Cover Classification of High-Resolution Imagery,“ *IEEE Geoscience and Remote Sensing Letters* , t. 14, br. 4, str. 549-553, 2017.

[235] C. Zhang, I. Sargent, X. Pan, H. Li, A. Gardiner, J. Hare i P. M. Atkinson, „An object-based convolutional neural network (OCNN) for urban land use classification,“ *Remote Sensing of Environment,* t. 216, str. 57-70, 2018.

[236] C. Zhang, X. Pan, H. Li, A. Gardiner, I. Sargent, J. Hare i P. M. Atkinson, „A hybrid MLP-CNN classifier for very fine resolution remotely sensed image classification,“ *ISPRS Journal of Photogrammetry and Remote Sensing*, t. 140, str. 133-144, 2018.

[237] P. Helber, B. Bischke, A. Dengel i D. Borth, „EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification,“ arXiv.org, 2019.

[238] X.-Y. Tong, G.-S. Xia, Q. Lu, H. Shen, S. Li, S. You i L. Zhang, „Learning Transferable Deep Models for Land-Use Classification with High-Resolution Remote Sensing Images,“ arXiv.org, 2018.

[239] S. S. Seferbekov, V. I. Iglovikov, A. V. Buslaev i A. A. Shvets, „Feature Pyramid Network for Multi-Class Land Segmentation,“ arXiv.org, 2018.

[240] W. Han, R. Feng, L. Wang i Y. Cheng, „A semi-supervised generative framework with deep learning features for high-resolution remote sensing image scene classification,“ *ISPRS Journal of Photogrammetry and Remote Sensing*, t. 145, str. 23-43, 2018.

[241] B. Jin, P. Ye, X. Zhang, W. Song i S. Li, „Object-Oriented Method Combined with Deep Convolutional Neural Networks for Land-Use-Type Classification of Remote Sensing Images,“ *Journal of the Indian Society of Remote Sensing*, t. 47, br. 6, str. 951-965, 2019.

[242] C. Zhang, J. Liu, F. Yu, S. Wan, Y. Han, J. Wang i G. Wang, „Segmentation model based on convolutional neural networks for extracting vegetation from Gaofen-2 images,“ *Journal of Applied Remote Sensing*, t. 12, br. 4, 2018.

[243] M. T. Torres, B. Perrat, M. Iliffe, J. Goulding i M. Valstar, „Automatic Pixel-Level Land-use Prediction Using Deep Convolutional Neural Networks,“ u *GISRUK 2017*, Manchester, 2017.

[244] R. Li, W. Liu, L. Yang, S. Sun, W. Hu, F. Zhang i W. Li, „DeepUNet: A Deep Fully Convolutional Network for Pixel-Level Sea-Land Segmentation,“ *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* , t. 11, br. 11, str. 3954-3962, 2018.

[245] O. Ronneberger, P. Fischer i T. Brox, „U-Net: Convolutional Networks for Biomedical Image Segmentation,“ arXiv.org, 2015.

[246] J. L. Gustafson, „Reevaluating Amdahl's law,“ *Communications of the ACM*, t. 31, br. 5, str. 532-533, 1988.

BIOGRAFIJA AUTORA

Igor Ševo je rođen u Banjoj Luci, 26. septembra 1990. godine gdje je upisao studije elektrotehnike i računarstva na Elektrotehničkom fakultetu Univerziteta u Banjoj Luci. Ove studije pohađao je od 2009. do 2013. godine. Završio je drugi ciklus studija na istom fakultetu i tu stekao zvanje magistra elektrotehnike i računarstva. Za vrijeme svog školovanja i studiranja učestvovao je u brojnim takmičenjima, od republičkog do svjetskog nivoa. Učestvovao je u mnogim projektima, uključujući i Horizon 2020 projekte, u kolaboraciji sa velikim brojem kompanija iz regiona i dalje, kao i u saradnji sa Elektrotehničkim fakultetom u Banjoj Luci. Radio je u obrazovnim institucijama kao nastavnik (profesor srednje škole u Gimnaziji u Banjoj Luci i asistent, kasnije i viši asistent, na Elektrotehničkom fakultetu Univerziteta u Banjoj Luci). Uporedo sa studijama i angažmanom u nastavi, samostalno je razvio i isporučio brojne softverske proizvode, te je radio na velikom broju eksperimentalnih projekata, uključujući pristupe automatskom generisanju muzičkih kompozicija, proceduralnom generisanju sadržaja, i mnogih drugih. Pored angažmana u primarnoj oblasti od interesa, i objavljenih radova u ovoj oblasti, napisao je i značajan broj muzičkih kompozicija, kao i diskusija i eseja na različite teme.

Прилог 3.

Изјава 1

ИЗЈАВА О АУТОРСТВУ

Изјављујем
да је докторска дисертација

Наслов рада Specijalizovana neuronska mreža za klasifikaciju i segmentaciju aero-snimaka

Наслов рада на енглеском језику Specialized neural network for aerial image classification and segmentation

- ☐ резултат сопственог истраживачког рада,
- ☐ да докторска дисертација, у цјелини или у дијеловима, није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- ☐ да су резултати коректно наведени и
- ☐ да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

У Бањој Луци 25.3.2020.

Потпис докторанта

Igor Šeno

Изјава 2

Изјава којом се овлашћује Универзитет у Бањој Луци
да докторску дисертацију учини јавно доступном

Овлашћујем Универзитет у Бањој Луци да моју докторску дисертацију под насловом

Specijalizovana neuronska mreža za klasifikaciju i segmentaciju aero-snimaka

која је моје ауторско дјело, учини јавно доступном.

Докторску дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у дигитални репозиторијум Универзитета у Бањој Луци могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (*Creative Commons*) за коју сам се одлучио/ла.

1. Ауторство
2. Ауторство – некомерцијално
3. Ауторство – некомерцијално – без прераде
4. Ауторство – некомерцијално – дијелити под истим условима
5. Ауторство – без прераде
6. Ауторство – дијелити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

У Бањој Луци 25.3.2020.

Потпис докторанта

Изјава 3

Изјава о идентичности штампане и електронске верзије докторске дисертације

Име и презиме аутора Igor Ševo

Наслов рада Specijalizovana neuronska mreža za klasifikaciju i segmentaciju aero-snimaka

Ментор dr Zdenka Babić

Изјављујем да је штампана верзија моје докторске дисертације идентична електронској верзији коју сам предао/ла за дигитални репозиторијум Универзитета у Бањој Луци.

У Бањој Луци 25.3.2020.

Потпис докторанта

Igor Ševo