Automatski dizajn konvolucijske neuronske mreže uporabom optimizacije rojem čestica

Ukić, Ante

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

Sveučilišni studij

AUTOMATSKI DIZAJN KONVOLUCIJSKE NEURONSKE MREŽE UPORABOM OPTIMIZACIJE ROJEM ČESTICA

Diplomski rad

Ante Ukić

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 15.02.2024.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Ante Ukić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1255R, 07.10.2021.
OIB studenta:	75245621337
Mentor:	doc. dr. sc. Dražen Bajer
Sumentor:	dr. sc. Mario Dudjak
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	doc. dr. sc. Krešimir Romić
Član Povjerenstva 1:	dr. sc. Mario Dudjak
Član Povjerenstva 2:	doc. dr. sc. Bruno Zorić
Naslov diplomskog rada:	Automatski dizajn konvolucijske neuronske mreže uporabom optimizacije rojem čestica
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	U radu je potrebno opisati problem automatskog dizajna konvolucijske neuronske mreže kao problem dvorazinske optimizacije. Uz to, potrebno je opisati metodu optimizacije rojem čestica kao i njezinu primjenu za rješavanje razmatranog problema. U praktičnom dijelu rada nužno je razviti programsko rješenje koje predstavlja inačicu metode optimizacije rojem čestica za rješavanje problema automatskog dizajna konvolucijske neuronske mreže. Također, potrebno je napraviti eksperimentalnu analizu na nekoliko poznatih problema klasifikacije iz literature. Sumentor: Mario Dudjak
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene od strane mentora:	15.02.2024.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 02.03.2024.

Ime i prezime studenta:	Ante Ukić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1255R, 07.10.2021.
Turnitin podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom**: Automatski dizajn konvolucijske neuronske mreže uporabom optimizacije rojem čestica**

izrađen pod vodstvom mentora doc. dr. sc. Dražen Bajer

i sumentora dr. sc. Mario Dudjak

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Duboko učenje (engl. *deep learning*) predstavlja područje umjetne inteligencije koje se bavi proučavanjem algoritama inspiriranih strukturom i funkcijama ljudskog mozga [1]. Umjetne neuronske mreže (engl. *artificial neural networks,* ANNs) jedne su od glavnih predstavnika modela dubokog učenja među kojima se ističu konvolucijske neuronske mreže (engl. *convolutional neural networks,* CNNs). Njihova primjena obuhvaća područja poput prepoznavanja objekata, analize slika, pretraživanja, medicinske dijagnostike te mnoge druge. Dizajn konvolucijskih neuronskih mreža predstavlja izazov čije rješavanje zahtijeva ulaganje znatnog napora i vremena. Dizajn mreža može se predstaviti kao optimizacijski problem u kojem je potrebno pronaći odgovarajuću strukturu mreže (određenu nizom parametara) za dani problem. Automatiziranjem dizajna CNNs ubrzao bi se proces pronalaska odgovarajuće strukture mreže za zadani problem. U literaturi su predložene brojne metode optimizacije koje nastoje riješiti problem automatskog dizajna mreže od kojih se algoritmi bio-inspiriranog računanja (engl. *bio-inspired computation*) nameću kao valjan izbor zbog svoje učinkovitosti. Jedan od predstavnika algoritama bio-inspiriranog računanja je algoritam optimizacije rojem čestica (engl. *particle swarm optimization,* PSO) koji se pokazao učinkovitim u rješavanju brojnih problema optimizacije, uključujući i dizajniranje neuronskih mreža. Osnovna ideja algoritma PSO proizlazi iz simuliranja ponašanja roja čestica u prostoru pretraživanja kako bi se pronašlo najbolje rješenje. Automatski dizajn mreže pomoću algoritma PSO može ubrzati i olakšati proces dizajna konvolucijskih neuronskih mreža, omogućavajući prilagodbu strukture mreže specifičnim zadacima i podacima.

Cilj ovog diplomskog rada je istražiti mogućnosti dizajna konvolucijskih neuronskih mreža korištenjem algoritma PSO. U ovom radu proučeni su različiti aspekti konvolucijskih neuronskih mreža, njihova struktura i funkcionalnost te će se detaljno istražiti algoritam PSO i njegova primjena u dizajnu neuronskih mreža. Dizajn konvolucijskih neuronskih mreža obuhvaća odabir arhitekture mreže koja je određena brojem slojeva, veličinom i vrstom slojeva koju ju čine. Provedbom eksperimentalne analize na skupovima podataka za prepoznavanje objekata i usporedbom s drugim pristupima optimizacije, ovaj rad će demonstrirati učinkovitost i prednosti automatskog dizajna konvolucijskih neuronskih mreža uporabom optimizacije rojem čestica. Dodatno će se u radu provesti ograničeno istraživanje utjecaja odabira parametara i načina predstavljanja rješenja u algoritmu PSO, kako bi se analizirao njihov utjecaj na izvedbu algoritma upotrjebljenog za automatski dizajn konvolucijskih neuronskih mreža.

U drugom poglavlju dan je uvod u konvolucijske neuronske mreže, opisana je struktura mreže te je predstavljen problem izgradnje mreže. Uz to, dan je kratki pregled literature s naglaskom na problem automatskog dizajna neuronskih mreža. U trećem poglavlju opisan je algoritam PSO. Predstavljeni su parametri algoritma te način na koji utječu na ponašanje algoritma. Opisani su različiti načini predstavljanja rješenja algoritma te je dan uvid u primjenu algoritma za automatski dizajn neuronskim mreža. U četvrtom poglavlju opisan je način rada izrađenog programskog rješenja te su uz to priloženi isječci koda. U petom poglavlju opisan je provedeni eksperiment i analizirani su rezultati algoritma PSO. Uspoređen je utjecaj načina predstavljanja rješenja te odabir vrijednosti parametara algoritma PSO. Uz to uspoređen je algoritam PSO s nekim od uobičajenih načina za dizajn neuronskih mreža. U šestom poglavlju dan je uvid u uspješnost provedene eksperimentalne analize i algoritma PSO za automatski dizajn neuronskih mreža te su navedene smjernice za budući rad.

2. KONVOLUCIJSKE NEURONSKE MREŽE

Konvolucijske neuronske mreže su jedne od poznatijih vrsta umjetnih neuronskih mreža koje se koriste za izradu modela iz raznih vrsta podataka. Korištenjem operacije konvolucije, CNNs su sposobne izdvajati značajke (engl. *features*) iz podataka čime se postiže viša razina apstrakcije. Pronalazak ispravne strukture CNN za modeliranje određenog problema može biti izazovan proces. Literatura ukazuje na složenost problema pronalaska strukture CNNs te daje uvid u dosadašnje pristupe rješavanja ovog problema.

2.1. Uvod u konvolucijske neuronske mreže

Umjetne neuronske mreže jedna su od temeljnih paradigmi u području umjetne inteligencije, a inspirirane su biološkim živčanim sustavom, prvenstveno mozgom (kod sisavaca) [2]. Modeli umjetnih neuronskih mreža mogu se shvatiti kao skup osnovnih procesnih jedinica koje su čvrsto povezane i djeluju na dane ulaze kako bi obradile informacije i generirale željene izlaze. U općem smislu, ANNs predstavljaju kompozicije raznih funkcija koje djeluju na linearnu kombinaciju ulaznih podataka [3]. Struktura ANN sastoji se od čvorova koji su predstavljeni tim funkcijama (analogno biološkim neuronima) i težinskim vezama koje povezuju čvorove (koje po analogiji odgovaraju sinapsama). Da bi se neka ANN primijenila na određeni problem, potrebno je dizajnirati odgovarajuću strukturu mreže [3][1]. Umjetne neuronske mreže mogu se grupirati u dvije kategorije na temelju načina na koji se informacije šire u mreži, a to su mreže s propagacijom signala prema naprijed (engl. *feed-forward neural network*) i mreže s propagacijom signala prema nazad (engl. *feed-back neural network*). Tok informacija u mreži s propagacijom signala prema naprijed odvija se samo u jednom smjeru. Ako se mreža smatra grafom s neuronima kao njegovim čvorovima, veze između čvorova su takve da nema petlji ili krugova u grafu. U ovu kategoriju se mogu svrstati konvolucijske neuronske mreže. U literaturi je predložen značajan broj različitih vrsti ANNs, a konvolucijske neuronske mreže karakteristične su po tome što same uče reprezentaciju značajki raznih vrsta ulaznih podataka kao što su slike, tekstovi, audiozapisi i slično.

Za razliku od ostalih neuronskih mreža, konvolucijske neuronske mreže koriste konvolucijski sloj na ulazu umjesto potpuno povezanog sloja. Mreža se može sastojati od više konvolucijskih slojeva koji koriste operaciju konvolucije kako bi otkrili značajke slika. Najraniji oblik konvolucijskih neuronskih mreža je model Neocognitron kojeg je predložio 1980. godine Kunihiko Fukushima [4]. Neocognitron je višeslojna umjetna neuronska mreža koja može prepoznati vizualne uzorke kroz učenje. Fukushima je bio inspiriran radom Davida Hubela i Torstena Wiesela koji su 1959. godine u svom radu naveli da su neuroni unutar mozga mačaka

posloženi u slojeve koji uče prepoznavati vizualne obrasce izdvajanjem osnovnih značajki i njihovim kombiniranjem kako bi se prikazala reprezentacija više razine [5].

2.2. Struktura mreže

Kao što je navedeno ranije, konvolucijske neuronske mreže se sastoje od više slojeva. Ova mreža se uobičajeno sastoji od tri vrste slojeva: konvolucijski sloj (engl. *convolutional layer*), sloj sažimanja (engl. *pooling layer*) te potpuno povezani sloj (engl. *fully connected dense layer*). Čvorovi ovih slojeva ugrađuju razne aktivacijske funkcije (engl. *activation function*) koje uvode nelinearnost u mrežu. Bez aktivacijskih funkcija mreža bi bila jednaka linearnom preslikavanju iz domene ulaza u domenu izlaza [6]. Neke od uobičajenih aktivacijskih funkcija su: sigmoidna (engl. *sigmoid*), ReLu (engl. *Rectified Linear Unit*) i linearna. U nastavku će biti navedeni i detaljno objašnjeni pojedini dijelovi konvolucijske neuronske mreže.

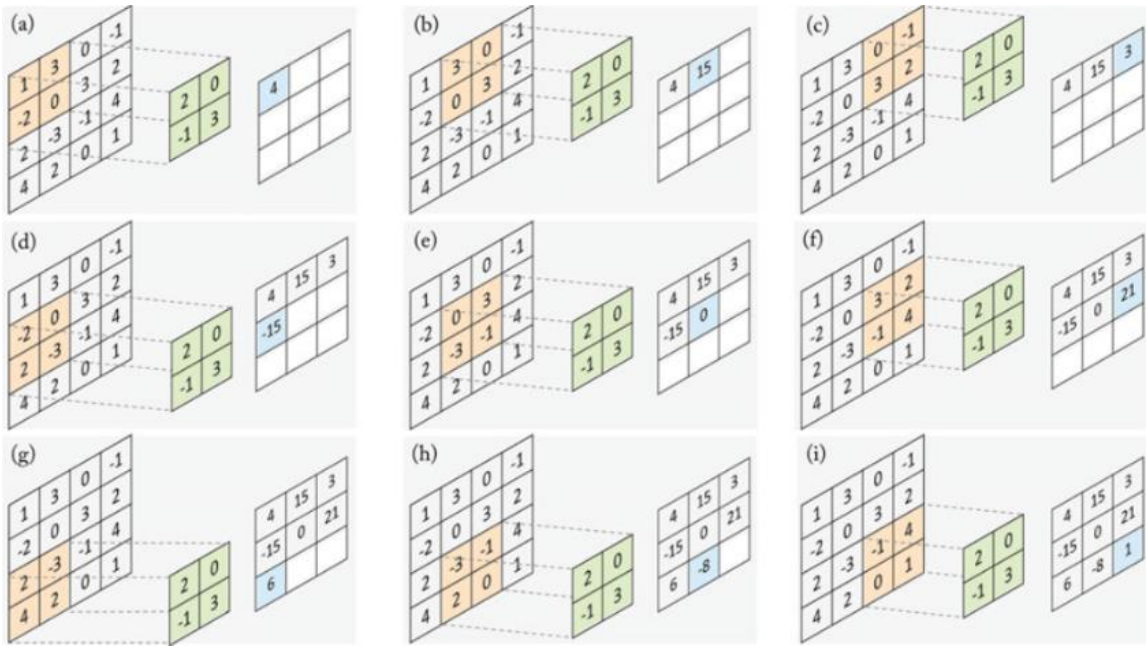
2.2.1. Konvolucijski sloj

Konvolucijski sloj je najvažniji sloj unutar konvolucijske neuronske mreže. Sastoji se od jezgre (engl. *kernel*) nad kojom se zajedno s ulazom vrši operacija konvolucije. Svaka jezgra unutar konvolucijskog sloja je mapa koja se sastoji od diskretnih brojeva. Veličina jezgre se postavlja prije treniranja, a neke od najčešćih veličina jesu 3x3, 5x5 i 7x7 [6]. Težina jezgre nije poznata prije početka treniranja nego se prilikom inicijalizacije generira nasumično te se kroz proces učenja težina jezgre mijenja. Mapa značajki (engl. *feature map*) predstavlja izlaz nakon primjene operacije konvolucije nad ulazom.

Ako za primjer uzmemo 2D mapu značajki veličine 4x4 i 2D jezgru veličine 2x2, konvolucijski sloj množi vrijednosti 2x2 jezgre s prvom 2x2 podmapom ulazne mape značajki te zbraja sve vrijednosti koja predstavlja prvi zapis u izlaznoj mapi značajki. Kako je prikazano na slici 2.1 jezgra se pomiče horizontalno i vertikalno po mapi značajki sve dok se ne popuni cijela izlazna mapa. Dimenzije izlazne mape se mogu izračunati kao

$$(m - n + 1) \times (m - n + 1), \tag{2.1}$$

gdje m predstavlja veličinu ulazne mape, a n veličinu jezgre.



Slika 2.1: *Prikaz operacije konvolucije nad mapom veličine 4x4 s jezgrom 2x2 [6]*

Korak kod konvolucijskih neuronskih mreža predstavlja iznos za koliko će se jezgra pomaknuti nad ulaznom mapom nakon izvršavanja operacije konvolucije. U prethodnom primjeru korak je bio 1 te se jezgra primjenjivala nad istim vrijednostima više puta. To se može izbjeći povećanjem koraka i samim time smanjiti veličinu izlazne mape. Nedostatak kod konvolucijskog sloja može biti gubitak podataka na rubovima. Kako bi se to izbjeglo može se primijeniti nadopunjavanje (engl. *padding*). Nadopunjavanje predstavlja postupak koji nadopunjuje nule na granicama ulaza. Nadopunjavanjem se kontrolira veličina izlazne mape. Korištenjem koraka i nadopunjavanja dimenzija izlazne mape računa se kao

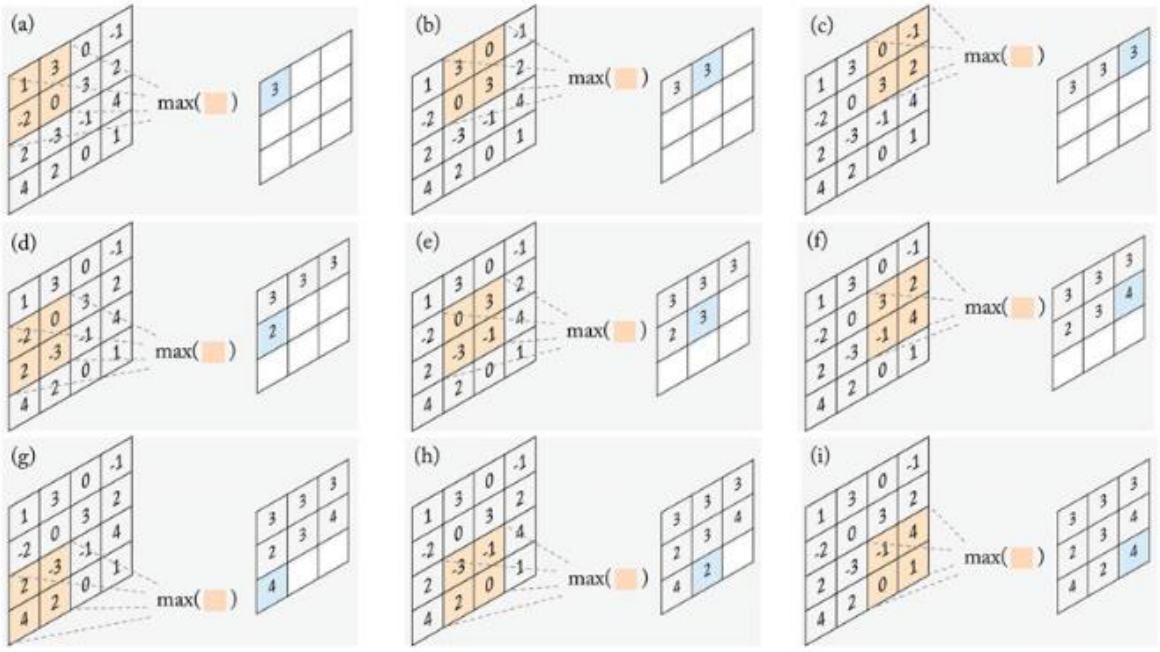
$$\left(\frac{m + 2 \times p - n}{S}\right) + 1 \times \left(\frac{m + 2 \times p - n}{S}\right) + 1, \tag{2.2}$$

gdje m predstavlja veličinu ulazne mape, n veličinu jezgre, p stupanj nadopunjavanje i S korak.

2.2.2. Sloj sažimanja

Sloj sažimanja služi kako bi se smanjila veličina mape značajki primjenom neke od operacija sažimanja. Jedne od poznatijih operacija sažimanja jesu sažimanje maksimalnih vrijednosti, sažimanje minimalnih vrijednosti i prosječno sažimanje. Kao i kod konvolucijskog sloja definiraju se jezgre i korak. Sažimanjem maksimalnih vrijednosti iz mape značajki obuhvaćene jezgrom odabire se maksimalna vrijednost i sprema se u izlaznu mapu. Primjenom prosječnog sažimanja se računa prosjek vrijednosti obuhvaćenih jezgrom te se sprema u izlaznu mapu. Kod sažimanja

minimalnih vrijednosti odabire se najmanja vrijednost i sprema u izlaznu mapu. Na slici 2.2 je prikazano maksimalno sažimanje nad 4x4 mapom s jezgrom 2x2.



Slika 2.2: *Primjena maksimalnog sažimanja [6]*

2.2.3. Potpuno povezani sloj

Kod konvolucijske neuronske mreže potpuno povezani sloj se najčešće nalazi na kraju mreže [6]. Svaki neuron u potpuno povezanom sloju je izravno povezan sa svim neuronima iz prethodnog sloja. Potpuno povezani sloj za ulaz prima mapu značajki iz zadnjeg sloja sažimanja ili konvolucijskog sloja i pretvara ju u vektor. Svrha potpuno povezanog sloja je interpretacija konačnih karakteristika iz prethodnih slojeva i donošenje odluka na temelju tih značajki. CNN može sadržavati više potpuno povezanih slojeva, posebno u dubljim arhitekturama kada višestruki slojevi omogućuju učenje složenijih značajki. Broj čvorova u potpuno povezanom sloju ovisi o problemu koji se rješava. Ako se koristi u svrhu klasifikacije, broj čvorova jednak je broju klasa [6]. Aktivacijska funkcija koja se često koristi u potpuno povezanom sloju je Softmax aktivacijska funkcija [7]. Softmax aktivacijska funkcija pretvara vrijednosti čvorova u vjerojatnosti koje ukazuju na pripadnost ulaza određenoj klasi.

2.3. Automatski dizajn mreže

2.3.1. Opis problema

Proces dizajna neuronskih mreža tradicionalno uključuje odabir strukture mreže te vrijednosti parametara u čvorovima. U kontekstu strojnog učenja, parametri strukture mreže ujedno

predstavljaju hiperparametre modela CNN. Izgradnja modela CNN podrazumijeva određivanje rasporeda i broja slojeva te hiperparametara koji definiraju ponašanje slojeva, poput veličine jezgre u konvolucijskom sloju, stupnja sažimanja u sloju sažimanja i slično. Prikladne vrijednosti hiperparametara modela CNN nije jednostavno odrediti te se one u pravilu zasebno utvrđuju za svaki problem. Problemi dubokog učenja često su vrlo složeni (u pogledu broja primjeraka, ali i intrinzične složenosti) pa izgradnja odgovarajućeg modela ne predstavlja lak zadatak. Obično je potrebno ispitati brojne vrijednosti hiperparametara radi utvrđivanja prikladne strukture mreže, što iziskuje značajan trud i utrošak vremena. Povrh toga, ručno ispitivanje ovih hiperparametara zamoran je i dugotrajan proces.

U literaturi su predloženi brojni postupci koji nastoje ubrzati i automatizirati izgradnju strukture neuronskih mreža. Ovi postupci u načelu rješavaju problem pretraživanja neuronske arhitekture (engl. *neural arhitecture search,* NAS). Glavni cilj je automatizirati ovaj proces izgradnje, omogućujući računalima da samostalno istraže prostor mogućih arhitektura i pronađu odgovarajuće strukture za specifične zadatke. Problem pretraživanja neuronske arhitekture obično definiraju njegove tri glavne komponente: prostor pretraživanja (engl. *search space*), metoda optimizacije (engl. *optimization method*) i metoda vrednovanja kandidata (engl. *candidate evaluation method*). Prostor pretraživanja predstavlja različite strukture mreže koje se mogu izgraditi i vrednovati za specifičan problem. Odabir ispravnog prostora pretraživanja može smanjiti složenost i ubrzati vrijeme pretraživanja. Kako bi se odabrao ispravan raspon poželjno je imati prethodno znanje o skupu podataka kako bi se razmatrale prikladne arhitekture za dani problem. Metoda optimizacije određuje kako istražiti prostor pretraživanja. Odabir ispravne metode optimizacije može značajno ubrzati proces pretrage i dati bolji krajnji rezultat. Metoda vrednovanja kandidata je odgovorna za usporedbu međurezultata tijekom faze pretrage. Samo vrednovanje je zahtjevan i dugotrajan proces koji se sastoji od izgradnje mreže, treniranja na odabranom skupu podataka i vrednovanja kako bi se prikazale performanse modela.

2.3.2. Pregled literature

Pretraživanje neuronske arhitekture je problem u koji je uložen značajan istraživački napor iz kojeg je proizašao niz postupaka za automatski dizajn arhitekture neuronskih mreža. Pristup odabiru prostora pretrage može se podijeliti na globalni prostor pretrage (engl. *global search space*) i na prostor pretrage temeljen na stanicama (engl. *cell-based search space*) [8]. Korištenjem globalnog prostora pretrage ne postavlja se puno ograničenja algoritmu. Dopušta se da algoritam odlučuje o vrstama slojeva, hiperparametara i veza između slojeva [9]. Prostor pretrage temeljen na stanicama ograničava algoritam u kreiranju proizvoljnih mreža. Korištenjem ovog pristupa

arhitektura se sastoji od manjih struktura koje se ponavljaju. Kao što je navedeno u prošlom poglavlju, odabir ispravne metode optimizacije može značajno ubrzati proces pretrage. Često korištene metode u literaturi su algoritmi bio-inspiriranog računanja, podržano učenje (engl. *reinforcement learning,* RL) i Bayesova optimizacija (engl. *Bayesian optimization*) [8]. Podržano učenje podrazumijeva upotrebu agenta koji korištenjem određene akcije komunicira s okolinom u svrhu dobivanja nagrade. Podržano učenje se može primijeniti na razne načine u svrhu rješavanja problema NAS. Najčešći pristup je da agentova akcija predstavlja postupak generiranja neuronske mreže [11]. U Bayesovoj optimizaciji koristi se model predikcije ciljne funkcije u svrhu odabira najboljeg argumenta koji će se vrednovati na stvarnoj funkciji. Primjenom Bayesove optimizacije za rješavanje problema NAS ciljna funkcija predstavlja performanse mreže, a argumenti njezinu arhitekturu [9]. Algoritmi bio-inspiriranog računanja uglavnom oponašaju biološke organizme koji žive u populacijama kao što su ptica, mravi i ribe. Jedinke razmjenjuju informacije te na osnovu toga prilagođavaju svoje ponašanje. U kontekstu problema NAS, populacija predstavlja različite arhitekture neuronske mreže. Jedinke se vrednuju kako bi im se pridružila kvaliteta te se pretraga obično blago usmjeruje oko onih najkvalitetnijih [8]. Ovakav pristup ima prednost nad ostalim načinima pretrage zbog toga što algoritmi bio-inspiriranog računanja primjenom lokalne i globalne pretrage imaju bolji uvid u prostor pretrage što može dovesti do boljih rezultata [12]. E. Real i S. Moore u svome radu [13] koristili su evolucijsku strategiju (engl. *evolutionary strategy*) za automatsko pretraživanje arhitektura neuronskih mreža kako bi smanjili ljudsko sudjelovanje u pretraživanju arhitektura neuronskih mreža. H. Pham i M. Guan koristili su genetski algoritam (engl. *genetic algorithm*) za pronalazak najbolje arhitekture mreže primjenjujući dijeljenje parametara (engl. *parameter sharing*) [14]. B. Wang i Y. Sun koristili su algoritam PSO za automatski dizajn konvolucijskih neuronskih mreža [15]. Primijenili su inačicu algoritma PSO koja dinamički mijenja veličinu kodiranog rješenja u svrhu lakše prilagodbe arhitekture konvolucijskih neuronskih mreža za problem klasifikacije slike.

3. OPTIMIZACIJA ROJEM ČESTICA

Automatski dizajn neuronskih mreža je zahtjevan problem čije rješavanje iziskuje puno truda i vremena. Za njegovo rješavanje postoje brojne metode optimizacije između kojih se ističu bio-inspirirani algoritmi. Bio-inspirirani algoritmi predstavljaju valjan izbor zbog mogućnosti usmjerenog pretraživanja te su u dosadašnjim primjenama pokazali dobre performanse. Stoga se u ovom radu proučava algoritam optimizacije rojem čestica kao jedan od predstavnika takvih algoritama te se ispituje za rješavanje problema NAS, konkretno za CNN.

3.1. Opis algoritma

PSO algoritam je bio-inspirirani optimizacijski algoritam koji se temelji na ponašanju roja koji su predložili J. Kennedy i R. Eberhart 1995. godine [16]. Algoritam simulira društveno ponašanje životinja koje žive u skupinama kao što su ptice. Način na koji funkcionira jato ptica značajno olakšava njihov život, poglavito u vidu potrage za hranom, vodom ili skloništem. Svaka od jedinki proučava ostatak skupine te na osnovnu njihovog ponašanja se prilagođava. Algoritam počinje s populacijom čestica (engl. *particles*) gdje svaka čestica predstavlja položaj u prostoru pretrage. U kontekstu dizajna neuronske mreže svaka čestica predstavlja jednu kombinaciju hiperparametara modela, odnosno predstavlja jednu strukturu neuronske mreže. U svakoj iteraciji se za svaku česticu određuje kvaliteta te se ovisno o rezultatu ažurira najbolji osobni položaj čestice (p_{best}). Nakon što se vrednuju sve čestice ažurira se najbolji globalni položaj cijele populacije (g_{best}). Kognitivna i socijalna komponenta omogućavaju čestici uvid u najbolji osobni položaj i najbolji globalni položaj te na osnovu toga donose odluke o smjeru kretanja. Kognitivna komponenta odnosi se na osobno iskustvo svake čestice i mogućnost pamćenja najboljeg osobnog položaja dok se socijalna komponenta odnosi na informacije koje čestica posjeduje o cijeloj populaciji i najboljem globalnom položaju cijele populacije [2]. Osim položaja čestice, ažurira se i brzina čestice (v). Nova brzina čestice ažurira se na temelju njezine trenutne brzine, razlike između njezinog najboljeg osobnog položaja i trenutnog položaja te razlike između globalnog najboljeg položaja i trenutnog položaja prema (3.1), gdje ω predstavlja inercijsku težinu (engl. *inertia weight*) čestice, a c_1 i c_2 koeficijente ubrzanja (engl. *acceleration coefficients*) za kognitivnu i socijalnu komponentu [17]. Vrijednosti inercijske težine i akceleracijskih koeficijenata predstavljaju parametre algoritma koji se postavljaju prije njegova korištenja. Nadalje, r_1 i r_2 su nasumično odabrani brojevi u intervalu [0,1], dok x predstavlja trenutni položaj čestice [17]. Ažuriranje brzine omogućava česticama da se drugačijom brzinom kreću prema boljim rješenjima

u prostoru pretrage. Nakon ažuriranja brzina računa se novi položaj čestice, kao što je izraženo prema (3.2) [16].

$$v_{ij} = \omega * v_{ij} + c_1 * r_1 * \left(p_{best_{ij}} - x_{ij}\right) + c_2 * r_2 * \left(g_{best_j} - x_{ij}\right) \tag{3.1}$$

$$x_{ij} = x_{ij} + v_{ij} \tag{3.2}$$

3.2. Postavljanje parametara

Algoritam PSO ima skup parametara koji utječu na njegovo ponašanje. Razumijevanje ovih parametara ključno je za uspješnu primjenu algoritma u različitim problemima optimizacije. Parametri kod PSO algoritma su inercijska težina (ω), akceleracijski koeficijenti (c_1 i c_2), veličina roja ili populacije (engl. *population size*) i broj iteracija.

Inercija čestice kontrolira količinu gibanja čestice uzimajući u obzir prethodnu brzinu čestice, odnosno kontrolira koliko će prethodni smjer kretanja čestice utjecati na novu brzinu. Inercija je izuzetno važna kako bi se osiguralo konvergentno ponašanje i ravnoteža između istraživanja (engl. *exploration*) i iskorištavanja (engl. *exploitation*) [2]. Kada je $\omega \geq 1$, brzine čestica ubrzano rastu s vremenom, što može dovesti do razilaženja roja. Kada je $\omega < 1$, čestice usporavaju dok njihova brzina ne padne na 0. Velike vrijednosti inercije potiču globalnu pretragu, a manje vrijednosti potiču lokalnu pretragu, no premale vrijednosti inercije mogu dovesti do toga da roj više nema mogućnost pretrage [2].

Veličina roja predstavlja broj jedinki u populaciji. Što je više jedinki u populaciji to je veća početna raznolikost populacije. S većom populacijom moguće je pretražiti veći dio prostora za pretraživanje sa svakom iteracijom. Uobičajeni broj čestica u roju je između 10 i 30 [2]. S prevelikim brojem čestica se povećava kompleksnost algoritma, a s premalim brojem čestica može doći do toga da prostor pretrage ostane neistražen. Odabir broja čestica ovisi i o problemu u kojem se algoritam primjenjuje.

Broj iteracija predstavlja izlazni uvjet kod PSO algoritma. Broj iteracija koje su potrebne da se postigne dobro rješenje ovisi o problemu za koji se PSO algoritam primjenjuje. Mali broj iteracija može dovesti do preranog zaustavljanja, a preveliki broj iteracija može dovesti do bespotrebnog trošenja resursa.

Akceleracijski koeficijenti kontroliraju utjecaj kognitivnih i socijalnih komponenti na ukupnu brzinu čestice. Još se nazivaju i parametri povjerenja pri čemu c_1 označava koliko čestica ima povjerenja u sebe, a c_2 koliko čestica ima povjerenja u svoje susjede [2]. Male vrijednosti

koeficijenata omogućavaju česticama da istražuju područja izvan dobrih regija prostora pretraživanja prije nego što budu privučene nazad. Visoke vrijednosti uzrokuju veće ubrzanje s naglim kretanjem prema dobrim regijama ili udaljavanjem od njih. Ako je $c_1 > 0, c_2 = 0$, svaka čestica pronalazi najbolju poziciju u svom susjedstvu što predstavlja lokalno pretraživanje, ali kada je $c_1 = 0, c_2 > 0$ cijeli roj se kreće prema najboljoj globalnoj poziciji u prostoru pretraživanja [2]. Koeficijenti se često postavljaju jednakima kako bi se postiglo jednako privlačenje prema lokalnom i globalnom najboljem rješenju [2].

3.3. Način predstavljanja rješenja

Oblik rješenja u PSO algoritmu ovisi o zadatku za koji se algoritam primjenjuje. Uobičajeni način predstavljanja rješenja podrazumijeva kodiranje svake čestice kao vektora realnih brojeva, gdje svaki element vektora odgovara jednoj varijabli u optimizacijskom problemu. Broj elemenata vektora odgovara dimenzionalnosti optimizacijskog problema. Često se događa da varijable u optimizacijskom problemu mogu poprimiti samo diskretne vrijednosti (primjerice, binarne ili cijele brojeve). Stoga je također čest način predstavljanja rješenja binarni, gdje je svaka čestica predstavljena vektorom binarnih brojeva [18]. Obično se čestice zapravo i dalje kodiraju kao vektori realnih brojeva, no prilikom vrednovanja se oni pretvaraju u binarne vrijednosti odgovarajućom metodom diskretizacije [18]. Kod cjelobrojnog načina prikazivanja rješenja svaka čestica prikazana je kao vektor, ali uz ograničenje da vrijednosti i pozicije čestica moraju biti cijeli brojevi, koje se uobičajeno ostvaruje diskretizacijom realnih vrijednosti u sklopu vrednovanja rješenja.

3.4. Primjena za automatski dizajn konvolucijske neuronske mreže

PSO algoritam može biti dobar odabir kao metoda optimizacije kod pristupanja problemu pretraživanja arhitekture neuronskih mreža (NAS) [15]. Algoritam PSO omogućuje usmjerenu pretragu u prostoru različitih arhitektura neuronskih mreža. Čestice predstavljaju pozicije u prostoru pretrage (odnosno strukture neuronskih mreža), a usmjereno pretraživanje operacijama algoritma PSO nastoji se postići brza konvergencija ka mrežama zadovoljavajuće izvedbe. PSO kombinira lokalno i globalno pretraživanje koristeći koncept lokalnog i globalnog najboljeg rješenja. Ovakav način pomaže izbjeći problem zaglavljivanja u lokalnim minimumima i omogućuje bolje istraživanje prostora pretrage. Učinkovitost PSO algoritma ovisi o odabiru njegovih parametara. Potrebno je pažljivo postaviti vrijednosti parametara kako bi se postigao željeni rezultat. Treniranje se provodi nakon inicijalizacije populacije čestica. Za svaku česticu kreira se model čija je struktura mreže opisana pozicijom čestice. Česticama se ažuriraju brzine i

pozicije u prostoru pretrage te se ovaj proces ponavlja sve dok algoritam ne dođe do izlaznog uvjeta.

Rješenja se kodiraju na način koji omogućuje predstavljanje različitih neuronskih arhitektura. Rješenja kodirana u obliku vektora predstavljaju najčešći oblik kodiranja kada se PSO primjenjuje u kontekstu pretraživanja strukture neuronske mreže. Vrijednosti hiperparametara uglavnom su diskretne (cjelobrojne) vrijednosti, za što je potrebno djelomično prilagoditi način ponašanja algoritma PSO. PSO algoritam je dizajniran za kontinuirane prostore pretrage te rad s diskretnim varijablama može dovesti do poteškoća. Uz visoku složenost problema, prostor pretrage može imati velik broj dimenzija, što otežava brzo i učinkovito pretraživanje prostora. Svaki element vektora predstavlja određeni parametar koji definira strukturu mreže, odnosno hiperparametar modela. Hiperparametri kod CNN mogu biti broj pojedinih slojeva mreže, veličina jezgre u konvolucijskom sloju i sloju sažimanja, broj značajki, broj čvorova, veličina jezgre, veličina mape značajki, iznos koraka i vrstu operacije sažimanja [15].

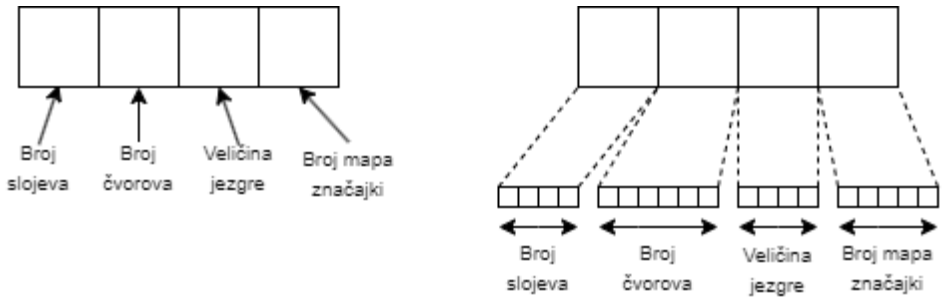
Funkcija cilja PSO algoritma izražava performanse neuronskih mreža koje se vrednuju tijekom pretrage. Točan odabir funkcije cilja ovisi o specifičnosti problema. Kod problema dubokog učenja, obično se kao funkcija cilja uzima prikladna mjera za iskazivanje performansi treniranog modela.

4. OSTVARENO PROGRAMSKO RJEŠENJE

U ovom poglavlju objašnjen je način rada programskog rješenja te su priloženi isječci koda uz kratke opise. Programsko rješenje implementirano je koristeći Python programski jezik zajedno s bibliotekama za strojno i duboko učenje. Korištene biblioteke su Keras [19] i scikit-learn [20]. Biblioteka Keras je korištena za implementaciju neuronskih mreža, dok se scikit-learn biblioteka koristila za evaluaciju performansi modela i podjelu skupa podataka na podskupove.

4.1. Način rada programskog rješenja

Implementacija PSO algoritma koristi se za pretragu prostora hiperparametara modela CNN. Odabrani hiperparametri korišteni za pretragu su broj skrivenih slojeva (engl. *hidden layers*) u mreži, broj čvorova (engl. *nodes*) unutar skrivenog sloja, veličina jezgre (engl. *kernel size*) i broj mapa značajki. Zbog ograničenosti računalnih resursa i vremena svaka mreža započinje s jednim konvolucijskim i jednim slojem sažimanja. Broj potpuno povezanih slojeva koji se postavljaju nakon prethodna dva sloja je promjenjiv. Time su mreže i dalje konvolucijske, a broj potpuno povezanih slojeva definira njihovu dubinu, odnosno složenost. Korištena su dva načina predstavljanja rješenja u algoritmu PSO. U prvoj implementaciji rješenja su vektori s 4 dimenzije u kojima svaki hiperparametar predstavlja jednu dimenziju vektora. Kod druge implementacije PSO algoritma rješenja predstavljaju vektore koji imaju 19 bitova. Vrijednost pojedinog hiperparametra je pretvorena u binarni oblik te njihovim spajanjem se kreira jedan binarni broj. U ovom slučaju prva 4 bita predstavljaju broj slojeva, sljedećih 6 bitova predstavljaju broj čvorova, 4 bita nakon toga predstavljaju veličinu jezgre te posljednjih 5 bitova predstavljaju broj mapa značajki.

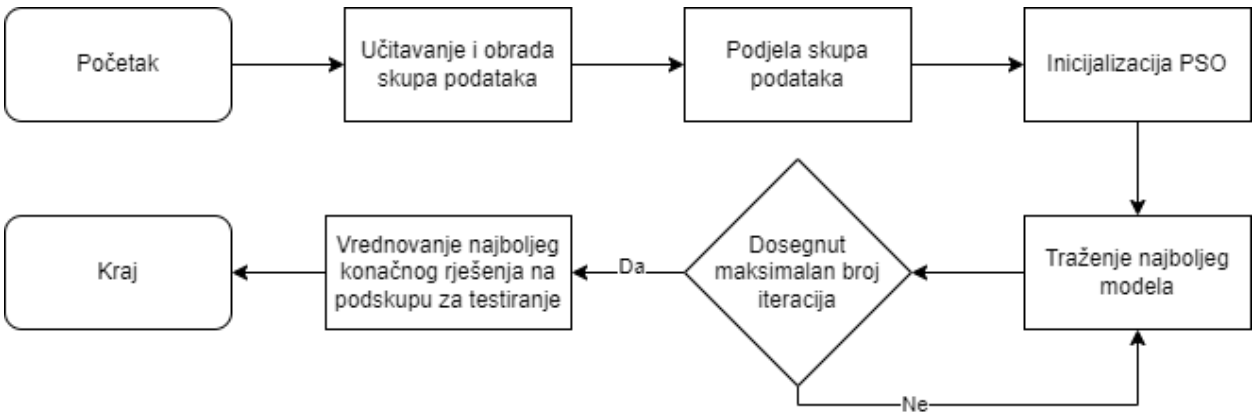


Slika 4.1: *Predstavljanje rješenja korištenih PSO implementacija*

Način na koji se čestice kreću kroz prostor pretrage može dovesti do toga da rješenja ne budu iz skupa prirodnih brojeva. Diskretizacija se postiže na način da se vrijednosti u vektoru rješenja zaokruže na najbližu cijelu vrijednost. Kod pokretanja PSO algoritma, početne vrijednosti čestica

su postavljene nasumično. Tijekom pretraživanja nova rješenja se uspoređuju s trenutno najboljim rješenjem na osnovi zadane funkcije cilja. Postavke programskog rješenja se mogu svesti na parametre za treniranje i hiperparametre. Parametri za treniranje su fiksno postavljeni unutar same implementacije neuronske mreže ili su postavljeni kao argumenti prilikom poziva funkcije te oni nisu kodirani u vektoru rješenja i ne mijenjaju se tijekom pretrage. Hiperparametri se definiraju u rasponu te se iz tog raspona uzimaju vrijednosti prilikom pretrage.

Programsko rješenje započinje s učitavanjem potrebnih biblioteka i skupa podataka. Skup podataka potrebno je oblikovati kako bi bio prikladan za treniranje modela. Skup podataka dijeli se na tri podskupa: podskup za treniranje, podskup za vrednovanje i podskup za testiranje. Podskup za treniranje koristi se kako bi model naučio odnose između ulaza i očekivanih izlaza. Podskup za vrednovanje služi za procjenu performansi modela tijekom pretrage. Podskup za testiranje koristi se za vrednovanje konačne performanse modela nakon pretrage. Potrebno je definirati metodu koja kreira model koji će se koristiti za treniranje i klasu koja predstavlja PSO algoritam. Unutar PSO klase definirane su metode za pretragu prostora pretraživanja, kreiranja nasumične čestice, računanja brzine čestice, ažuriranja pozicije čestice u prostoru pretraživanja, evaluacije modela i statičke metode koja će ažurirati najbolje globalno rješenje. Pokretanjem programskog rješenja pretražuje se prostor pretraživanja dok se ne dosegne maksimalan broj iteracija nakon čega se kao rezultat pretrage vraća najbolja mreža. Pretraživanje započinje inicijalizacijom populacije čestica. Za svaku česticu kreira se model s odgovarajućom strukturom konvolucijske mreže. Model se zatim trenira na podskupu za treniranje te se njegove performanse vrednuju na podskupu za vrednovanje. Nakon toga, za svaku česticu se računa brzina, ažurira se položaj čestice te se trenira i vrednuje model sve dok se ne dosegne maksimalan broj iteracija. Tijekom pretraživanja u zasebnu datoteku zapisuje se kvaliteta i matrica zabune od svakog novog najboljeg rješenja, kod problema klasifikacije. U konačnici, na osnovu najboljeg pronađenog rješenja izgrađuje se model CNN koji se zatim vrednuje na podskupu za testiranje.



Slika 4.2: *Dijagram toka programskog rješenja*

4.2. Prikaz i način uporabe programskog rješenja

U ovom poglavlju su priloženi isječci programskog koda za pronalaženje najbolje strukture neuronske mreže korištenjem PSO algoritma. Na slici 4.3 je prikazan način na koji se učitava skup podataka te se podaci unutar skupa preoblikuju kako bi bili prikladni za treniranje modela.

```
(X_train_full, y_train_full), (X_test, y_test) = mnist.load_data()
X_train_full = X_train_full.reshape(X_train_full.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
X_train_full = X_train_full.astype('float32') / 255
X_test = X_test.astype('float32') / 255
y_train_full = to_categorical(y_train_full, 10)
y_test = to_categorical(y_test, 10)
```

Slika 4.3: *Učitavanje i obrada skupa podataka*

Slika 4.4 prikazuje funkciju create_model koja se koristi za kreiranje modela za treniranje. Funkcija za parametre prima kombinaciju hiperparametara i fiksne parametre potrebne za kreiranje slojeva. Funkcija kreira model koji predstavlja konvolucijsku neuronsku mrežu. Mreža se sastoji od jednog konvolucijskog sloja i jednog sloja sažimanja. Broj skrivenih slojeva i čvorova unutar skrivenog sloja ovisi o predanim parametrima.

```
def create_model(layers=1, nodes=1, kernel_size=3, conv_stride=1, feature_maps=3, pool_size=3, pool_stride=3, pool_type='average'):
    model = Sequential()
    model.add(Conv2D(feature_maps, kernel_size=(kernel_size, kernel_size), strides=(conv_stride, conv_stride),
                     input_shape=(28, 28, 1), kernel_initializer=glorot_normal(), activation=relu))
    model.add(AveragePooling2D(pool_size=(pool_size, pool_size), strides=(pool_stride, pool_stride), padding='valid'))
    model.add(Flatten())
    for i in range(layers):
        model.add(Dense(nodes, activation=relu))
        model.add(Dropout(0.5))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])
    return model
```

Slika 4.4: *Funkcija za kreiranje modela*

```
conv_stride = 1
pool_size = 3
pool_stride = 2
epochs = 3
epochs_for_best = 20
batch_size = 200
runs = 10
bounds = {
    "min_bounds": [1, 250, 3, 3],  # redom za layers, nodes, kernel_sizes, feature_maps
    "max_bounds": [10, 300, 7, 32]  # redom za layers, nodes, kernel_sizes, feature_maps
}
```

Slika 4.5: *Definiranje fiksnih parametara i raspon hiperparametara*

Slika 4.6 prikazuje PSO klasu i konstruktor PSO klase. Na slikama 4.7 i 4.8 su prikazane dvije metode koje pripadaju PSO klasi. Metoda sa slike 4.7 evaluira performanse na validacijskom skupu podataka. Kao parametar prima trenutnu kombinaciju hiperparametara koje prosljeđuje funkciji za kreiranje modela. Model se evaluira i njegove performanse se zapisuju. Na slici 4.8 se nalazi implementacija metode za pretragu. Ova metoda je ključan dio PSO algoritma te služi kako bi se pronašla najbolja kombinacija hiperparametara.

```
class ParticleSwarmOptimisation:
    def __init__(self, population_size, max_nfes, c1, c2, inertia, bounds, X_train, y_train, X_val, y_val):
        self.start_time = None
        self.conv_stride = None
        self.batch_size = None
        self.epochs = None
        self.pool_stride = None
        self.pool_size = None
        self.create_model_function = None
        self.population_size = population_size
        self.max_nfes = max_nfes
        self.c1 = c1
        self.c2 = c2
        self.inertia = inertia
        self.bounds = bounds
        self.layers = None
        self.nodes = None
        self.kernel_size = None
        self.feature_maps = None
        self.min_bounds = bounds["min_bounds"]
        self.max_bounds = bounds["max_bounds"]
        self.encoding = {
            "layers_idx": 0,
            "nodes_idx": 1,
            "kernel_size_idx": 2,
            "feature_maps_idx": 3
        }
        self.dimensionality = len(self.encoding.keys())
        self.X_train = X_train
        self.y_train = y_train
        self.X_val = X_val
        self.y_val = y_val
```

Slika 4.6: *Konstruktor klase PSO*

```
def _evaluate_on_validation(self, candidate):
    model = KerasClassifier(model=self.create_model_function, epochs=self.epochs, batch_size=self.batch_size,
                            verbose=0,
                            validation_split=0,
                            layers=int(round(candidate[self.encoding["layers_idx"]], 0)),
                            nodes=int(round(candidate[self.encoding["nodes_idx"]])),
                            kernel_size=int(round(candidate[self.encoding["kernel_size_idx"]])),
                            conv_stride=self.conv_stride,
                            feature_maps=int(round(candidate[self.encoding["feature_maps_idx"]])),
                            pool_size=self.pool_size,
                            pool_stride=self.pool_stride)
    model.fit(self.X_train, self.y_train)
    predictions = model.predict(self.X_val)
    f1 = f1_score(self.y_val, predictions, average='macro')
    cm = confusion_matrix(self.y_val.argmax(axis=1), predictions.argmax(axis=1))
    logging.info("Fitness: " + str(f1))
    logging.info("Confusion matrix: " + str(cm))
    return f1
```

Slika 4.7: Metoda za vrednovanje rješenja čestice

```
def search(self, create_model_function, pool_size, pool_stride, epochs, batch_size, conv_stride):
    self.create_model_function = create_model_function
    self.pool_size = pool_size
    self.pool_stride = pool_stride
    self.epochs = epochs
    self.batch_size = batch_size
    self.conv_stride = conv_stride
    spent_nfes = 0
    self.start_time = time.time()
    particle_positions = np.empty((self.population_size, self.dimensionality), float)
    particle_fitness = np.zeros(self.population_size, float)
    particle_velocities = np.zeros((self.population_size, self.dimensionality), float)
    particle_personal_best = np.empty((self.population_size, self.dimensionality), float)
    particle_personal_best_fitness = np.zeros(self.population_size)
    particle_global_best = np.empty(self.dimensionality)
    particle_global_best_fitness = -1
    for i in range(self.population_size):
        particle_positions[i] = self._create_random_particle()
        particle_fitness[i] = self._evaluate_on_validation(particle_positions[i])
        particle_personal_best[i] = copy.deepcopy(particle_positions[i])
        particle_personal_best_fitness[i] = particle_fitness[i]
        if particle_personal_best_fitness[i] > particle_global_best_fitness or particle_global_best_fitness == -1:
            particle_global_best = copy.deepcopy(particle_personal_best[i])
            particle_global_best_fitness = particle_personal_best_fitness[i]
        spent_nfes += 1
        if spent_nfes / self.max_nfes in log_points:
            logging.info("Global best fitness: " + str(particle_global_best_fitness))
    while spent_nfes < self.max_nfes:
        for i in range(self.population_size):
            particle_velocities[i] = self._calculate_velocity(i, particle_velocities, particle_positions,
                                                              particle_personal_best, particle_global_best)
            particle_positions[i] = self._update_particle(i, particle_positions, particle_velocities)
            particle_fitness[i] = self._evaluate_on_validation(particle_positions[i])
            if particle_fitness[i] > particle_personal_best_fitness[i]:
                particle_personal_best[i] = copy.deepcopy(particle_positions[i])
                particle_personal_best_fitness[i] = particle_fitness[i]
            spent_nfes += 1
            if spent_nfes >= self.max_nfes:
                break
            else:
                if spent_nfes / self.max_nfes in log_points:
                    logging.info("Spent nfes: " + str(spent_nfes))
                    logging.info("Global best fitness: " + str(particle_global_best_fitness))
                    logging.info("Layers: {}, Nodes: {}, Kernel size: {}, Feature maps: {}".format(particle_global_best[0],
                    particle_global_best[1], particle_global_best[2], particle_global_best[3]))
        particle_global_best, particle_global_best_fitness = self._update_global_best(
            particle_global_best, particle_global_best_fitness,
            particle_personal_best_fitness, particle_personal_best)
        if spent_nfes >= self.max_nfes:
            logging.info("Global best fitness: " + str(particle_global_best_fitness))
    self.layers = particle_global_best[0]
    self.nodes = particle_global_best[1]
    self.kernel_size = particle_global_best[2]
    self.feature_maps = particle_global_best[3]
    duration = time.time() - self.start_time
    return particle_global_best, particle_global_best_fitness
```

Slika 4.8: *Metoda za traženje najboljeg rješenja*

Slike 4.9 i 4.10 prikazuju ispis iz datoteke za zapisivanje. Slika 4.9 predstavlja ispis koji za zapisuje nakon svake evaluacije, a slika 4.10 prikazuje ispis koji se zapisuje nakon pronalaska novog najboljeg globalnog rješenja.

```
INFO:root:Fitness: 0.9803298420833384
INFO:root:Confusion matrix:
[[1165    0    1    0    1    0    2    2    4    0]
 [   0 1309    8    0    2    0    1    1    1    0]
 [   0    3 1162    2    0    0    0    2    4    1]
 [   1    0   22 1178    0    5    0    3    8    2]
 [   2    1    0    0 1165    0    1    2    3    2]
 [   4    0    0    3    1 1081    4    0   11    0]
 [   4    0    1    0    1    1 1166    0    4    0]
 [   0    5   11    2    3    0    0 1275    2    1]
 [   2    0    6    2    6    2    3    2 1136    1]
 [   6    2    1    1   19    3    0   20   14 1128]]
```

Slika 4.9: *Ispis u datoteku za zapisivanje nakon evaluacije*

```
INFO:root:Global best fitness: 0.9872981121286241
INFO:root:Best network has 5 layers, 300 nodes within layers,
7 kernel size and 32 feature maps
INFO:root:Test F1: 0.992132835803862
INFO:root:Test confusion matrix:
[[ 976    1    0    0    0    0    1    1    1    0]
 [   0 1130    0    1    1    1    1    0    1    0]
 [   1    0 1022    0    1    0    0    8    0    0]
 [   0    0    0 1004    0    3    0    0    1    2]
 [   0    0    0    0  976    0    2    0    1    3]
 [   0    0    0    6    0  882    1    1    1    1]
 [   3    2    0    0    1    1  950    0    1    0]
 [   0    1    3    2    0    0    0 1021    1    0]
 [   4    0    4    1    0    1    1    0  962    1]
 [   0    1    0    0    4    2    0    1    2  999]]
```

Slika 4.10: *Ispis u datoteku za zapisivanje nakon evaluacije najboljeg modela*

5. EKSPERIMENTALNA ANALIZA I REZULTATI

Kako bi se utvrdila uspješnost algoritma PSO za rješavanje problema automatskog dizajna konvolucijskih neuronskih mreža potrebno je provesti eksperimentalnu analizu. Eksperimentalna analiza se može podijeliti na četiri dijela. U prvom dijelu analizira se utjecaj načina predstavljanja rješenja algoritma. U drugom dijelu analizira se utjecaj vrijednosti parametara algoritma. U trećem dijelu eksperimentalne analize najbolja verzija algoritma PSO (u smislu odabranih vrijednosti parametara te načina predstavljanja rješenja) uspoređuje se s nekoliko uobičajenih postupaka za automatski dizajn neuronskih mreža kako bi se stekao općenitiji uvid u uspješnost algoritma PSO. U četvrtom dijelu algoritam PSO uspoređuje se s algoritmom AutoKeras koji predstavlja potpuno automatsko rješenje za dizajn neuronskih mreža.

5.1. Postavke i metodologija eksperimentalne analize

Eksperimentalna analiza je provedena na četiri različita skupa podataka kako bi se mogle usporediti performanse i rezultati u ovisnosti o različitim ulaznim podacima. Korišteni skupovi podataka su MNIST, fashion MNIST, CIFAR-10 i CIFAR-100 [19]. Skup podataka MNIST sadrži skup slika na kojima se nalaze rukom pisani brojevi. MNIST se sastoji od 70000 crno-bijelih slika veličine 28 x 28 piksela podijeljenih u 10 klasa po 7000 slika gdje svaka klasa predstavlja jedan broj. Fashion MNIST skup sadrži 70000 crno-bijelih slika veličine 28 x 28 piksela koje su podijeljene u 10 klasa od kojih svaka klasa predstavlja drukčiji odjevni predmet. Ukupan broj slika podijeljen je po klasama te svakoj klasi pripada 7000 slika. Skup CIFAR-10 se sastoji od 60000 slika u boji koje su veličine 32 x 32 piksela. Slike su podijeljene u 10 različitih klasa po 6000 slika. CIFAR-100 je prošireni CIFAR-10 skup. Isto kao i skup CIFAR-10, CIFAR-100 se sastoji od 60000 slika u boji veličine 32 x 32 piksela. Razlika između ova dva skupa je što je skup CIFAR-100 podijeljen na 100 različitih klasa raspodijeljenih da svaka klasa sadrži 600 slika.

Eksperimentalna analiza započinje podjelom korištenih skupova podataka. Svi skupovi podataka podijeljeni su u podskup za treniranje i podskup za testiranje. Kod skupova podataka MNIST i Fashion MNIST 60000 slika je postavljeno u skup za treniranje, a 10000 slika u skup za testiranje. Kod skupova podataka CIFAR-10 i CIFAR-100 50000 slika je postavljeno u skup za treniranje, a 10000 slika u skup za testiranje. Podskup za treniranje dodatno je podijeljen u omjeru 80:20% gdje se 80% podskupa koristi za treniranje modela, a 20% podskupa za vrednovanje modela tijekom treniranja. Podjela skupa podataka je nasumična kako bi se izbjeglo ponavljanje sličnih podskupova. Kako bi se ublažio utjecaj stohastičke prirode algoritama i da bi se dobio općenitiji uvid u njihove performanse eksperiment je proveden 10 puta. Nakon podjele skupova

podataka postavljaju se parametri algoritma. Kako je zadatak ovog rada automatski dizajn konvolucijske mreže parametri se mogu podijeliti na parametre koji su fiksno zadani te neće biti mijenjani tijekom treniranja te na hiperparametre koji su zadani u određenom rasponu i prema kojima će algoritmi tražiti najbolju strukturu mreže. Za sve algoritme parametri će biti jednaki. Fiksni parametri su navedeni u tablici 5.1.

Tablica 5.1: *Fiksni parametri i zadane vrijednosti za eksperiment*

Parametri	Vrijednosti
Aktivacijska funkcija	ReLu
Inicijalizacijska težina	Xavier
Algoritam za treniranje mreže	Adam
Stopa učenja	0.001
Veličina serije	200
Stopa odbacivanja	0.5
Broj epoha za treniranje rješenja tijekom pretrage	3
Broj epoha za treniranje konačnog rješenja	200
Mjera za vođenje treniranja mreže	Kategorička unakrsna entropija
Pomak jezgre	1
Pomak pri sažimanju	2
Operacija sažimanja	Prosječno sažimanje

Hiperparametri i njihov raspon vrijednosti su navedeni u tablici 5.2.

Tablica 5.2: *Hiperparametri i raspon vrijednosti za eksperiment*

Hiperparametri	Minimalna vrijednost	Maksimalna vrijednost
Broj (potpuno povezanih) slojeva	1	10
Broj čvorova u sloju	250	300
Veličina jezgre	3	7
Broj mapa značajki	3	33

Također je potrebno postaviti spomenute parametre algoritma PSO. Postavke parametara iz tablice 5.3 su inicijalno postavljene kako bi se mogao usporediti način predstavljanja rješenja te će se u nastavku usporediti s još 5 različitih postavki parametara. Broj iteracija algoritma PSO je postavljen na 100 što predstavlja 3000 vrednovanja rješenja (100 * Ns).

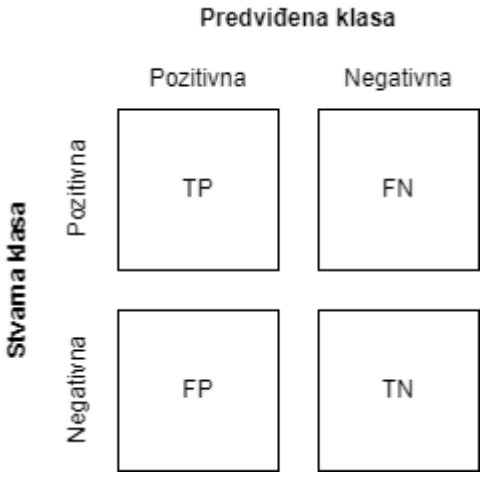
Tablica 5.3: *Vrijednosti parametara PSO algoritma*

Parametri	Maksimalna vrijednost
Ns	30
ω	0.724
c1, c2	1.468

Funkcija cilja u algoritmu PSO predstavljena je mjerom F1, koja je uobičajena mjera za vrednovanje izvedbe klasifikacijskog modela [21]. F1 mjera može poprimiti vrijednosti u intervalu [0,1]. Mjera F1 se računa kao

$$F_1 = 2 * \left(\frac{TP}{2 * TP + FN + FP}\right), \tag{5.1}$$

gdje TP predstavlja broj ispravnih pozitivnih rezultata, FN predstavlja broj neispravnih negativnih rezultata i FP predstavlja broj neispravnih pozitivnih rezultata, a izvode se iz matrice zabune (prikazane na slici 5.1).



Slika 5.1: *Matrica zabune*

5.2. Rezultati i diskusija

Rezultati su podijeljeni u zasebne tablice po skupovima podataka. Za svaki skup podataka prikazane su performanse najboljih pronađenih mreža za svako izvođenje eksperimenta. Najbolji rezultati za pojedino izvođenje su podebljani te je na dnu tablice prikazana prosječna i medijalna vrijednost rezultata te standardna devijacija (σ).

5.2.1. Utjecaj načina predstavljanja rješenja

Kao što je objašnjeno ranije, postoji više načina na koje se može predstaviti rješenje PSO algoritma. U ovom radu uspoređena su dva načina predstavljanja rješenja, odnosno kodiranje rješenja pomoću vektora realnih ili binarnih vrijednosti. Oba načina su pojašnjena u četvrtom poglavlju. Prvi način (kodiranje pomoću vektora realnih vrijednosti) u tablicama je označen kao PSO, dok je drugi način (kodiranje pomoću vektora binarnih vrijednosti) označen kao BPSO.

U tablicama 5.4, 5.5, 5.6 i 5.7 prikazane su performanse najboljih pronađenih mreža na podskupu za testiranje izražene mjerom F1. Iz prikazanih rezultata može se vidjeti kako je algoritam PSO znatno uspješniji kada se koristi prvi način predstavljanja rješenja. Treba podsjetiti da je dimenzionalnost prostora pretrage u prvom načinu predstavljanja rješenja znatno manja (četiri elementa u odnosu na 19). U tablici 5.8 prikazan je omjer najboljih rezultata između dva načina kodiranja. Prvi način predstavljanja rješenja je imao bolje rezultate u 90% slučajeva (u 36 od 40 izvođenja eksperimenta). U tablici 5.9 prikazan je Pearsonov koeficijent korelacije između kvaliteta najboljih mreža na podskupovima za vrednovanje i testiranje. Iz tablice 5.9 može se vidjeti da je korelacija između podskupova za vrednovanje i testiranje veća kod algoritma PSO. Iz slike 5.2 može se vidjeti kako kvaliteta rješenja ima nagli rast u ranijim fazama pretrage te se pred kraj stabilizira i ima blagi rast. U početku pretrage rješenja se generalno ne razlikuju, no kasnije algoritam PSO pronalazi kvalitetnija rješenja.

Tablica 5.4: *Usporedba performansi PSO i BPSO algoritama na MNIST skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO	BPSO
MNIST	1.	**0.9921**	0.9886
	2.	**0.9915**	0.9842
	3.	**0.9929**	0.9878
	4.	**0.9917**	0.9899
	5.	0.9917	**0.9920**
	6.	**0.9924**	0.9907
	7.	**0.9905**	0.9898
	8.	0.9913	**0.9919**
	9.	**0.9925**	0.9898
	10.	**0.9923**	0.9885
	$\overline{F1}$	**0.9919**	0.9893
	σ	**0.0007**	0.0023
	F_{med}	**0.9919**	0.9898

Tablica 5.5: *Usporedba performansi PSO i BPSO algoritama na Fashion MNIST skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO	BPSO
Fashion MNIST	1.	**0.9207**	0.9138
	2.	**0.9160**	0.9131
	3.	**0.9163**	0.8901
	4.	**0.9200**	0.8991
	5.	**0.9198**	0.8821
	6.	**0.9186**	0.9074
	7.	**0.9203**	0.9119
	8.	**0.9172**	0.9123
	9.	**0.9175**	0.9087
	10.	**0.9163**	0.9117
	$\overline{F1}$	**0.9183**	0.9050
	σ	**0.0018**	0.0110
	F_{med}	**0.9180**	0.9102

Tablica 5.6: *Usporedba performansi PSO i BPSO algoritama na CIFAR-10 skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO	BPSO
CIFAR - 10	1.	**0.6569**	0.6242
	2.	**0.6709**	0.6618
	3.	**0.6591**	0.6549
	4.	**0.6633**	0.6316
	5.	**0.6680**	0.6464
	6.	**0.6561**	0.6449
	7.	**0.6518**	0.6302
	8.	0.6646	**0.6664**
	9.	**0.6757**	0.6303
	10.	**0.6786**	0.6309
	$\overline{F1}$	**0.6645**	0.6422
	σ	**0.0088**	0.0149
	F_{med}	**0.6640**	0.6382

Tablica 5.7: *Usporedba performansi PSO i BPSO algoritama na CIFAR-100 skupu podataka*

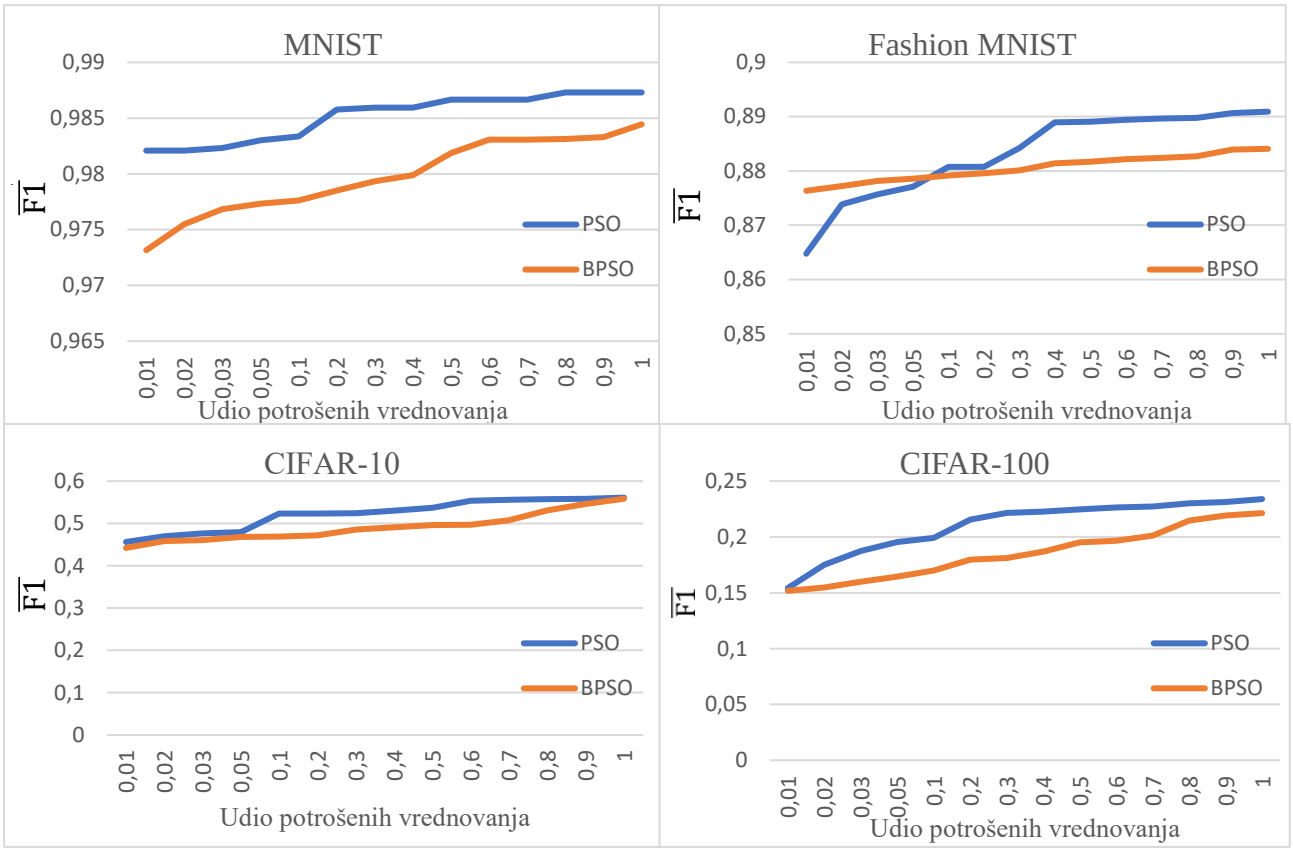
Skup podataka	Izvođenje eksperimenta	PSO	BPSO
CIFAR - 100	1.	**0.3327**	0.3230
	2.	**0.3112**	0.3060
	3.	0.3076	**0.3139**
	4.	**0.3358**	0.3025
	5.	**0.3546**	0.3322
	6.	**0.3457**	0.3323
	7.	**0.3373**	0.3277
	8.	**0.3238**	0.3078
	9.	**0.3172**	0.3043
	10.	**0.3214**	0.3129
	$\overline{F1}$	**0.3287**	0.3163
	σ	0.0151	**0.0116**
	F_{med}	**0.3283**	0.3134

Tablica 5.8: *Sažete performanse algoritama PSO i BPSO na podskupovima za testiranje*

Skup podataka	PSO		BPSO	
	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$
MNIST	**8**	**0.9919**	2	0.9893
Fashion MNIST	**10**	**0.9183**	0	0.9050
CIFAR – 10	**9**	**0.6645**	1	0.6422
CIFAR – 100	**9**	**0.3287**	1	0.3163

Tablica 5.9: *Pearsonov koeficijent korelacije algoritama PSO i BPSO između kvaliteta najboljih mreža na podskupovima za vrednovanje i testiranje*

Skup podataka	PSO	BPSO
MNIST	**0.9344**	0.8577
Fashion MNIST	**0.9361**	0.8609
CIFAR – 10	**0.9584**	0.9287
CIFAR – 100	**0.9397**	0.9296



Slika 5.2: *Kvalitete najboljih rješenja tijekom pretrage*

5.2.2. Utjecaj odabira vrijednosti parametara

Ponašanje PSO algoritma i način na koji se čestice kreću kroz prostor pretrage ovisi o parametrima, kao što je ranije objašnjeno. Kako bi se pronašla najbolja kombinacija parametara potrebno je ispitati različite postavke parametara algoritma PSO. Ispitane postavke parametara algoritma preuzete su iz [22], a preuzeto je 6 postavki parametara koje su prikazane u tablici 5.10.

Tablica 5.10: *Ispitane postavke parametara algoritma PSO*

	ω	c_1	c_2
PSO-2	0.729	2.0412	0.9477
PSO-3	0.6	1.7	1.7
PSO-6	0.724	1.468	1.468
PSO-8	0.837	1.255	1.255
PSO-11	0.5	1.90	1.90
PSO-14	-0.1	0.875	2.625

Kod PSO-2 postavki postavljena je veće vrijednost c_1 koeficijenta što znači da je algoritam imao jaču orijentaciju prema najboljem lokalnom rješenju što može uzrokovati brže lokalno

konvergiranje. To može rezultirati manjom raznolikošću i slabijem globalnom rješenju. PSO-6 postavke imaju umjerenu vrijednost inercijske težine i ravnomjernu raspodjelu između lokalne i globalne pretrage. Kod PSO-3 i PSO-11 postavki akceleracijski koeficijenti imaju jednaku vrijednost što podupire ravnotežu između lokalnog i globalnog pretraživanja ali vrijednost inercijske težine je niska što dovodi do sporijeg istraživanja prostora pretrage. PSO-8 postavke imaju visoku vrijednost inercijske težine i niske vrijednosti akceleracijski koeficijenata. S takvim parametrima algoritam je pristraniji globalnom istraživanju. Postavke PSO-14 imaju negativnu vrijednost inercijske težine što može dovesti do nepredvidljivog ponašanja. Uz to niska vrijednost c_1 koeficijenta i visoka vrijednost c_2 koeficijenta dovodi do pristranosti prema globalnim optimumima.

U tablicama 5.11, 5.12, 5.13 i 5.14 su prikazani rezultati uspoređenih verzija algoritma PSO za pojedini skup podataka izraženi mjerom F1. Generalno najbolje rezultate daje algoritam PSO-6 ($\omega = 0.724$, $c_1 = 1.468$, $c_2 = 1.468$). Kroz četiri skupa podataka PSO-6 postavke su imale najbolji rezultat u 37.5% slučajeva (u 15 od 40 izvođenja eksperimenta) te su na tri skupa podataka imale najveću prosječnu vrijednost funkcije cilja. Algoritam PSO-6 daje najbolje rezultate zbog toga što ima ravnotežu između lokalne i globalne pretrage uz umjerenu inercijsku težinu što omogućava istraživanje različitih dijelova prostora pretrage. U tablici 5.16 prikazan je Pearsonov koeficijent korelacije između kvaliteta najboljih mreža na podskupovima za validaciju i testiranje. Generalno svi algoritmi imaju visoku korelaciju između kojih se ističe algoritam PSO-8. Iz slike 5.3 vidi se kako početne vrijednosti najboljih rješenja imaju podjednaku vrijednost na skupu podataka MNIST, ali se krajnje vrijednosti razlikuju dok na skupovima podatak CIFAR-10 i CIFAR-100 imaju podjednake vrijednosti i na početku i na kraju pretrage.

Tablica 5.11: *Usporedba performansi različitih inačica PSO na MNIST skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO-2	PSO-3	PSO-6	PSO-8	PSO-11	PSO-14
MNIST	1.	0.9921	0.9920	0.9921	0.9912	0.9916	**0.9922**
	2.	**0.9928**	0.9915	0.9915	0.9919	0.9903	0.9917
	3.	0.9923	0.9924	**0.9929**	0.9923	0.9905	0.9921
	4.	0.9901	0.9904	**0.9917**	0.9912	0.9916	0.9915
	5.	0.9920	0.9922	0.9917	0.9908	**0.9924**	0.9916
	6.	0.9911	0.9914	**0.9924**	0.9917	0.9923	0.9899
	7.	0.9924	0.9909	0.9905	0.9916	0.9917	**0.9926**
	8.	0.9920	0.9920	0.9913	0.9921	**0.9927**	0.9915
	9.	0.9914	0.9920	**0.9925**	0.9923	0.9908	0.9923
	10.	0.9908	0.9924	0.9923	0.9910	0.9914	**0.9929**
	$\overline{F1}$	0.9917	0.9917	**0.9919**	0.9916	0.9916	0.9915
	σ	0.0008	0.0007	0.0007	**0.0005**	0.0008	0.0008
	F_{med}	**0.9920**	**0.9920**	0.9919	0.9917	0.9917	0.9917

Tablica 5.12: *Usporedba performansi različitih inačica PSO na Fashion MNIST skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO-2	PSO-3	PSO-6	PSO-8	PSO-11	PSO-14
Fashion MNIST	1.	0.9143	0.9142	**0.9207**	0.9161	0.9149	0.9201
	2.	0.9051	0.9160	0.9160	0.9141	0.9156	**0.9171**
	3.	0.9146	0.9138	0.9163	**0.9176**	0.9168	0.9155
	4.	0.9172	0.9175	**0.9200**	0.9190	0.9148	0.9160
	5.	0.9185	0.9136	**0.9198**	0.9041	0.9193	0.9180
	6.	0.9153	0.9154	0.9186	**0.9193**	0.9176	0.9186
	7.	0.9200	0.9180	**0.9203**	0.9085	0.9182	0.9139
	8.	0.9135	0.9161	**0.9172**	0.9171	0.9149	0.9165
	9.	0.9127	0.9112	0.9175	0.9147	0.9078	**0.9182**
	10.	0.9187	0.9150	0.9163	**0.9188**	0.9113	0.9143
	$\overline{F1}$	0.9150	0.9151	**0.9183**	0.9149	0.9151	0.9168
	σ	0.0043	0.0020	**0.0018**	0.0050	0.0034	0.0020
	F_{med}	0.9149	0.9152	**0.9180**	0.9166	0.9153	0.9168

Tablica 5.13: *Usporedba performansi različitih inačica PSO na CIFAR-10 skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO-2	PSO-3	PSO-6	PSO-8	PSO-11	PSO-14
CIFAR-10	1.	0.6685	0.6498	0.6569	0.6650	**0.6760**	0.6560
	2.	0.6633	0.6315	0.6709	0.6376	0.6530	**0.6758**
	3.	**0.6690**	0.6572	0.6591	0.6599	0.6665	0.6599
	4.	**0.6755**	0.6542	0.6633	0.6528	0.6608	0.6603
	5.	0.6587	**0.6813**	0.6680	0.6763	0.6332	0.6794
	6.	0.6505	0.6554	0.6561	**0.6697**	0.6532	0.6612
	7.	0.6507	0.6673	0.6518	0.6518	**0.6769**	0.6558
	8.	0.6492	0.6558	**0.6646**	0.6587	0.6644	0.6642
	9.	0.6547	0.6629	**0.6757**	0.6560	0.6288	0.6368
	10.	0.6685	0.6514	**0.6786**	0.6339	0.6566	0.6542
	$\overline{F1}$	0.6609	0.6567	**0.6645**	0.6562	0.6569	0.6603
	σ	0.0094	0.0128	**0.0088**	0.0132	0.0160	0.0118
	F_{med}	0.6610	0.6556	**0.6640**	0.6573	0.6587	0.6601

Tablica 5.14: *Usporedba performansi različitih inačica PSO na CIFAR-100 skupu podataka*

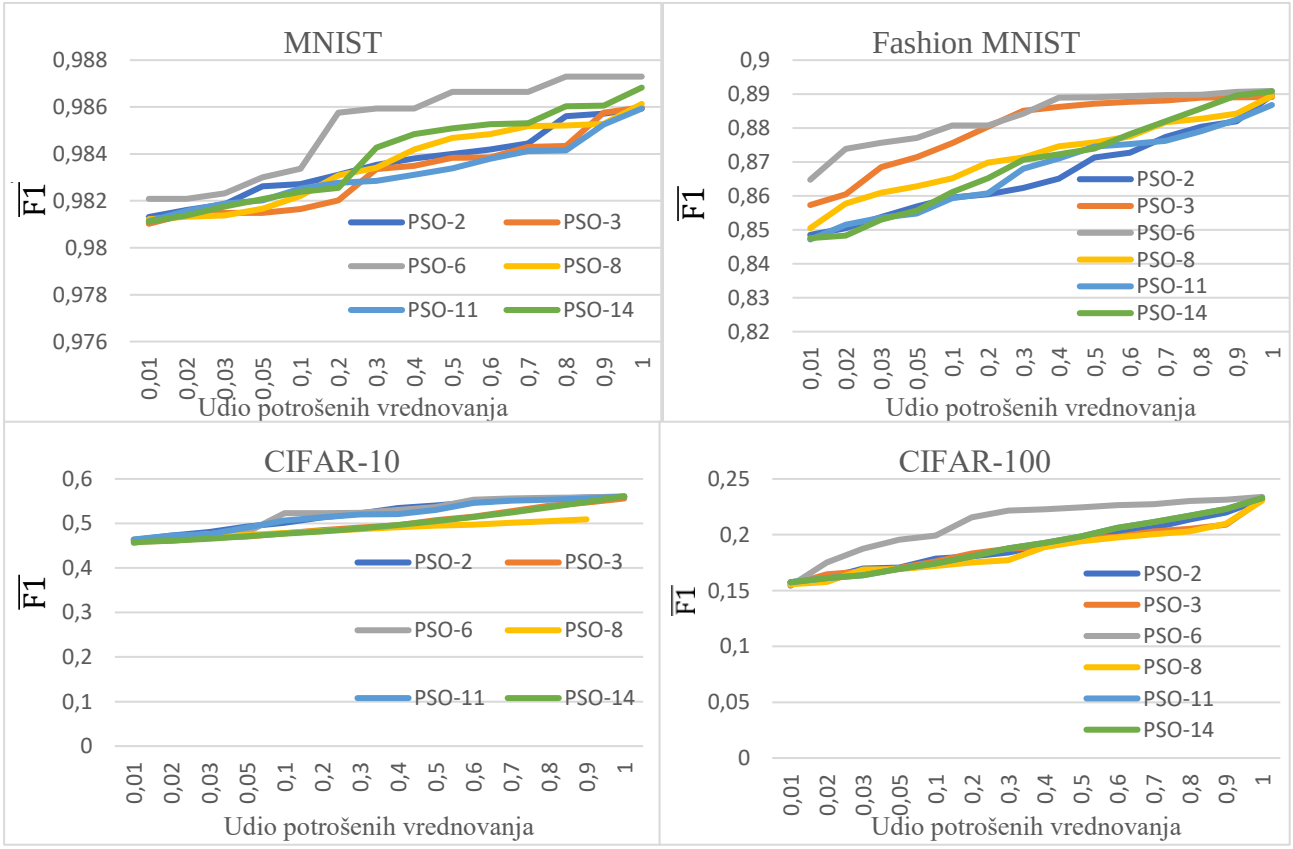
Skup podataka	Izvođenje eksperimenta	PSO-2	PSO-3	PSO-6	PSO-8	PSO-11	PSO-14
CIFAR-100	1.	0.3072	**0.3354**	0.3327	0.3025	0.3102	0.3316
	2.	0.3027	**0.3261**	0.3112	0.2867	0.3253	0.3236
	3.	0.3296	0.3392	0.3076	**0.3457**	0.3330	0.3286
	4.	0.3255	0.2915	**0.3358**	0.3005	0.3248	0.3314
	5.	0.3263	0.3360	**0.3546**	0.3211	0.3366	0.3319
	6.	0.3301	0.2888	**0.3457**	0.3290	0.3400	0.3277
	7.	0.3061	0.2569	0.3373	0.3189	**0.3420**	0.3377
	8.	0.3288	0.3088	0.3238	0.2960	**0.3479**	0.3400
	9.	0.2585	0.3414	0.3172	**0.3492**	0.3049	0.3181
	10.	0.2845	0.3042	0.3214	**0.3369**	0.3282	0.3207
	$\overline{F1}$	0.3099	0.3128	0.3287	0.3187	**0.3293**	0.3291
	σ	0.0236	0.0279	0.0151	0.0217	0.0137	**0.0070**
	F_{med}	0.3163	0.3174	0.3283	0.3200	**0.3306**	0.3300

Tablica 5.15: *Sažete performanse inačica PSO algoritma*

Skup podataka	PSO-2		PSO-3		PSO-6		PSO-8		PSO-11		PSO-14	
	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$
MNIST	1	0.9917	0	0.9917	**4**	**0.9919**	0	0.9916	2	0.9916	3	0.9915
Fashion MNIST	0	0.9150	0	0.9151	**5**	**0.9183**	3	0.9149	0	0.9151	2	0.9168
CIFAR-10	2	0.6609	1	0.6567	**3**	**0.6645**	1	0.6562	2	0.6569	1	0.6603
CIFAR-100	0	0.3099	2	0.3128	**3**	0.3287	**3**	0.3187	2	**0.3293**	0	0.3291

Tablica 5.16: *Pearsonov koeficijent korelacije između kvaliteta najboljih mreža na podskupovima za vrednovanje i testiranje*

Skup podataka	PSO-2	PSO-3	PSO-6	PSO-8	PSO-11	PSO-14
MNIST	0.8968	0.4203	0.9344	0.9138	**0.9587**	0.8822
Fashion MNIST	0.9046	0.9102	0.9361	**0.9413**	0.9582	0.9266
CIFAR-10	0.9023	0.9389	0.9584	**0.9770**	0.9565	0.9172
CIFAR-100	0.8296	0.9218	**0.9397**	0.9371	0.9299	0.9592



Slika 5.3: *Kvalitete najboljih rješenja različitih inačica PSO algoritma tijekom pretrage*

5.2.3. Usporedba s uobičajenim načinima za podešavanje hiperparametara

Kako bi se mogla potvrditi uspješnost algoritma PSO, performanse i rezultati su uspoređeni s nekoliko uobičajenih (i jednostavnijih) algoritama za podešavanje hiperparametara. Za usporedbu su se koristile dvije verzije algoritma nasumičnog pretraživanja (engl. *random search,* RS) i algoritam iscrpnog pretraživanja (engl. *exhaustive search,* ES). Prva verzija nasumičnog algoritma pretraživanja prije svakog vrednovanja potpuno nasumično odabire kombinaciju hiperparametara iz zadanih raspona. Druga verzija nasumičnog pretraživanja prvo nasumično generira 3000 kombinacija hiperparametara te nakon toga redom prolazi kroz sve te kombinacije. Prva verzija u tablicama je označena s RS, a druga s RGS. Algoritam iscrpnog pretraživanja nastoji ispitati sve kombinacije hiperparametara, ali se prekine nakon 3000 vrednovanja kako bi se pravedno usporedio s ostalim algoritmima.

Iz rezultata u tablicama 5.17, 5.18, 5.19 i 5.20 je vidljivo kako je algoritam PSO postigao znatno bolje rezultate u odnosu na uobičajene algoritme za automatski dizajn CNNs. Neovisno o podatkovnom skupu, PSO algoritam ima najbolju prosječnu vrijednost F1 mjere. PSO algoritam u 62.5% slučajeva ima najbolje performanse ako se uzme u obzir pokretanje eksperimenta na svim podatkovnim skupovima. To potvrđuje i tablica 5.20 u kojoj je prikazan broj najboljih rezultata za svaki algoritam po podatkovnom skupu. Treba napomenuti kako su početni parametri, raspon hiperparametara i broj vrednovanja bili jednaki za sve algoritme. Uspješnost algoritma PSO nad ostalim algoritmima može se pripisati načinu na koji pretražuje prostor. Algoritam PSO usmjereno pretražuje prostor pretrage, dok ostali algoritmi pretražuju nasumično ili unaprijed zadanim redom. U tablici 5.22 prikazane su prosječne vrijednosti i standardne devijacije hiperparametara najboljih mreža. Iz tablice se vidi kako algoritam ES pretražuje mreže s početnim vrijednostima hiperparametara zbog ograničenog broja vrednovanja. Algoritmi PSO, RS i RGS imaju približno jednake prosječne vrijednosti broja slojeva, mapi značajki i veličine jezgre, dok broj čestica znatno varira. Sa slike 5.4 vidi se da algoritam PSO ima generalno najmanju oscilaciju performansi. Ostali algoritmi imaju velika odstupanja od medijana.

Tablica 5.17: *Usporedba performansi PSO algoritma i uobičajenih algoritama za podešavanje hiperparametara na MNIST skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO	ES	RS	RGS
MNIST	1.	0.9921	0.9904	**0.9930**	0.9920
	2.	0.9915	0.9912	0.9917	**0.9922**
	3.	**0.9929**	0.9914	0.9913	0.9906
	4.	0.9917	**0.9923**	0.9912	0.9921
	5.	**0.9917**	0.9913	0.9911	0.9895
	6.	**0.9924**	0.9912	0.9920	0.9882
	7.	0.9905	**0.9925**	0.9910	0.9913
	8.	0.9913	0.9913	**0.9921**	0.9919
	9.	**0.9925**	0.9906	0.9908	0.9913
	10.	**0.9923**	0.9917	0.9909	0.9902
	$\overline{F1}$	**0.9919**	0.9914	0.9915	0.9909
	σ	**0.0007**	**0.0007**	**0.0007**	0.0013
	F_{med}	**0.9919**	0.9913	0.9913	0.9913

Tablica 5.18: *Usporedba performansi PSO algoritma i uobičajenih algoritama za podešavanje hiperparametara na Fashion MNIST skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO	ES	RS	RGS
Fashion MNIST	1.	**0.9207**	0.9002	0.9117	0.9157
	2.	0.9160	0.9122	**0.9161**	0.9057
	3.	0.9163	0.9027	0.9142	**0.9176**
	4.	**0.9200**	0.9120	0.9110	0.9076
	5.	**0.9198**	0.9124	**0.9198**	0.9035
	6.	0.9186	**0.9237**	0.9065	0.9094
	7.	**0.9203**	0.9024	0.9126	0.9129
	8.	**0.9172**	0.9117	0.9139	0.9101
	9.	**0.9175**	0.9121	0.9022	0.9135
	10.	**0.9163**	0.9012	0.9087	0.9154
	$\overline{F1}$	**0.9183**	0.9091	0.9117	0.9111
	σ	**0.0018**	0.0073	0.0050	0.0047
	F_{med}	**0.9180**	0.9119	0.9122	0.9115

Tablica 5.19: *Usporedba performansi PSO algoritma i uobičajenih algoritama za podešavanje hiperparametara na CIFAR-10 skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO	ES	RS	RGS
CIFAR-10	1.	0.6569	0.6445	0.6499	**0.6760**
	2.	**0.6709**	0.6308	0.6281	0.6597
	3.	**0.6591**	0.6380	0.6259	0.6323
	4.	0.6633	0.6406	0.6359	**0.6652**
	5.	**0.6680**	0.6452	0.6395	0.6564
	6.	**0.6561**	0.6325	0.6501	0.6536
	7.	0.6518	0.6454	**0.6611**	0.6455
	8.	**0.6646**	0.6403	0.6436	0.6353
	9.	**0.6757**	0.6418	0.6218	0.6082
	10.	**0.6786**	0.6475	0.6656	0.6444
	$\overline{F1}$	**0.6645**	0.6406	0.6422	0.6477
	σ	0.0088	**0.0055**	0.0147	0.0192
	F_{med}	**0.6640**	0.6412	0.6416	0.6496

Tablica 5.20: *Usporedba performansi PSO algoritma i uobičajenih algoritama za podešavanje hiperparametara na CIFAR-100 skupu podataka*

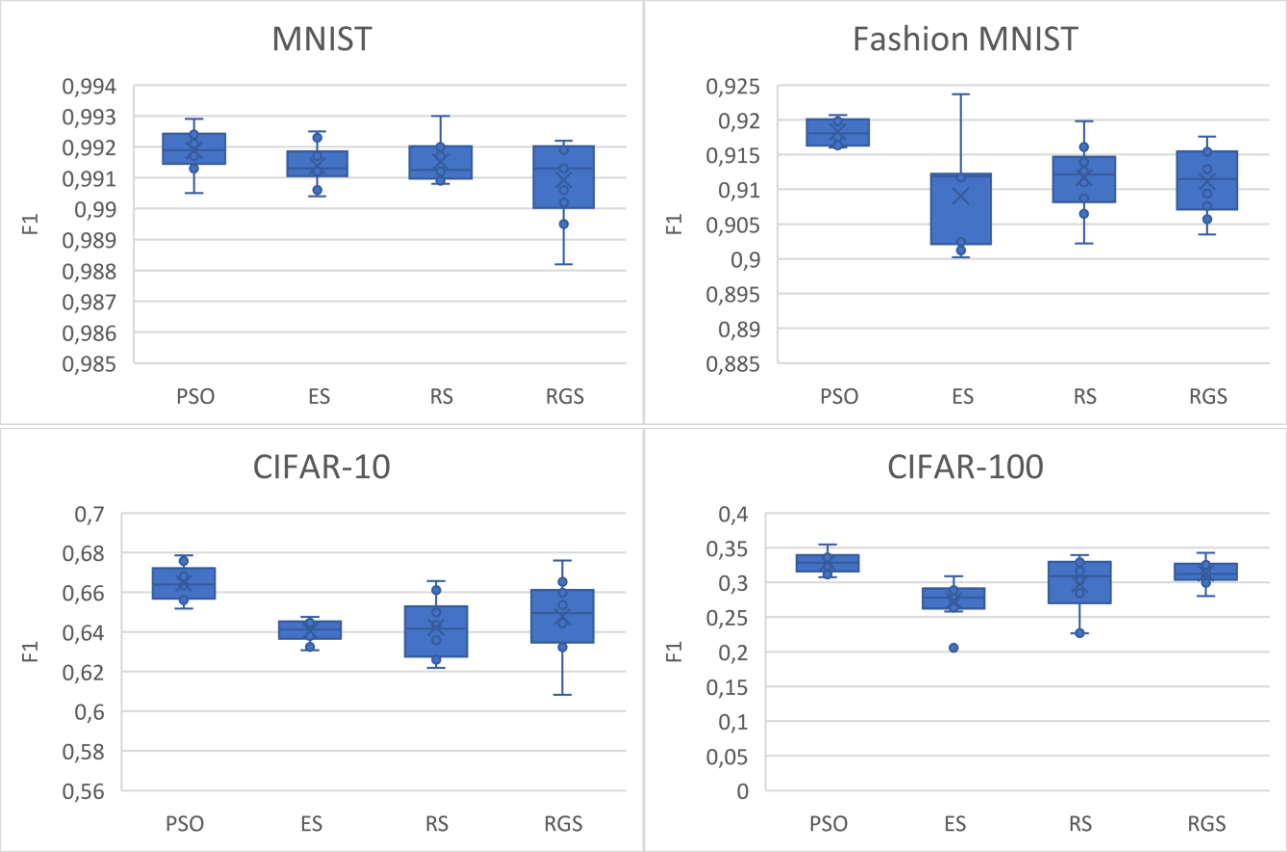
Skup podataka	Izvođenje eksperimenta	PSO	ES	RS	RGS
CIFAR-100	1.	**0.3327**	0.2732	0.2847	0.3110
	2.	0.3112	0.2789	**0.3317**	0.3131
	3.	**0.3076**	0.2581	0.2264	0.2802
	4.	0.3358	0.2852	**0.3394**	0.2995
	5.	**0.3546**	0.3087	0.3098	0.3251
	6.	**0.3457**	0.2979	0.2268	0.3050
	7.	**0.3373**	0.2634	0.3160	0.3059
	8.	0.3238	0.2889	0.3075	**0.3423**
	9.	**0.3172**	0.2054	0.3031	0.3171
	10.	0.3214	0.2767	0.3287	**0.3318**
	$\overline{F1}$	**0.3287**	0.2736	0.2974	0.3131
	σ	**0.0151**	0.0283	0.0405	0.0175
	F_{med}	**0.3283**	0.2778	0.3087	0.3120

Tablica 5.21: *Sažete performanse PSO algoritma i uobičajenih algoritama za podešavanje hiperparametara*

Skup podataka	PSO		ES		RS		RGS	
	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$
MNIST	**5**	**0.9919**	2	0.9914	2	0.9915	1	0.9909
Fashion MNIST	**7**	**0.9183**	1	0.9091	1	0.9117	1	0.9111
CIFAR-10	**7**	**0.6645**	0	0.6406	1	0.6422	2	0.6477
CIFAR-100	**6**	**0.3287**	0	0.2736	2	0.2974	2	0.3131

Tablica 5.22: *Prikaz prosječnih vrijednosti i standardne devijacije hiperparametara najboljih mreža*

Skup podataka		PSO				ES				RS				RGS			
		Slojevi	Čestice	Jezgra	Značajke	Slojevi	Čestice	Jezgra	Značajke	Slojevi	Čestice	Jezgra	Značajke	Slojevi	Čestice	Jezgra	Značajke
MNIST	$\overline{N}$	3.9	285.5	6.7	30.8	1	264.4	4.5	21.52	3.3	264.1	5.9	24.9	3.4	282.3	5	22.3
	σ	0.74	18.20	0.48	2.1	0	3.75	0.53	1.89	1.42	16.36	1.1	4.89	1.58	5.74	1.76	6.25
Fashion MNIST	$\overline{N}$	1.7	282.4	4	27.9	1	258.3	5.5	24.8	1.8	278.5	5.4	22.1	2	273	4.8	21.1
	σ	0.67	15.2	0	6.45	0	2.74	0.75	2.5	0.92	14.19	1.35	7.59	0.82	13.78	1.62	6.05
CIFAR-10	$\overline{N}$	1.2	291.2	3.9	29	1	261.6	6.2	20.3	1.4	276.9	4.3	19.4	2	274.2	5.3	21.8
	σ	0.42	10.16	0.74	5.33	0	4.88	0.8	4.2	0.52	14.63	1.42	7.11	0.94	19.56	1.34	7.36
CIFAR-100	$\overline{N}$	1.2	281.4	5.3	25.5	1	263.7	5.8	25.82	1.5	269.7	5.4	20.3	1.2	277.9	5	20.5
	σ	0.63	18.13	0.95	6.19	0	3.4	0.67	3.7	0.7	14.3	1.35	9.21	0.42	15.71	1.7	3.89



Slika 5.4: *Usporedba performansi najboljih mreža pronađenih algoritmom PSO i uobičajenim algoritmima za podešavanje hiperparametara*

5.2.4. Usporedba s AutoKeras algoritmom

Predloženi postupak za rješavanje problema pronalaska arhitekture neuronske mreže zahtijeva znanje o arhitekturama mreže i utjecaju hiperparametara i njihovih vrijednosti na arhitekturu mreže. Algoritam AutoKeras pojednostavljuje proces pronalaska arhitekture neuronske mreže automatskim pretraživanjem hiperparametara i arhitekture mreže. AutoKeras algoritam razvio je DataLab na državnom sveučilištu u Texasu [23]. AutoKeras podržava različite zadatke strojnog učenja, uključujući klasifikaciju slika, regresiju, klasifikaciju teksta i strukturnih podataka. Algoritam PSO uspoređuje se s AutoKeras algoritmom zato što AutoKeras predstavlja gotovu opciju za rješavanje problema automatskog dizajna neuronskih mreža koja ne zahtijeva ručno postavljanje raspona hiperparametara. AutoKeras algoritam kao metodu optimizacije primjenjuje Bayesovu optimizaciju. Kako bi usporedba bila što ispravnija, eksperimentalna analiza s AutoKeras algoritmom pokrenuta je na isti način kao što je opisano u poglavlju 5.1.

Iz rezultata prikazanih u tablicama 5.23, 5.24, 5.25 i 5.26 algoritam PSO ima znatno bolje rezultate na skupovima podataka MNIST i fashion MNIST, dok AutoKeras ima neznatno bolje rezultate na skupovima CIFAR-10 i CIFAR-100. Prema tome možemo zaključiti kako su trenutne postavke algoritma PSO uspješne u radu s jednostavnijim skupovima podataka, dok AutoKeras ima bolju

mogućnost prilagodbe kompleksnijim skupovima što daje nešto bolje rezultate u tom slučaju. Na slici 5.5 vidi se kako rezultati oba algoritma ne odstupaju puno od medijalne vrijednosti na skupovima podataka MNIST i Fashion MNIST, dok su na skupovima podataka CIFAR-10 i CIFAR-100 velika odstupanja od medijalne vrijednosti.

Tablica 5.23: *Usporedba performansi PSO i AutoKeras algoritama na MNIST skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO	AutoKeras
MNIST	1.	**0.9921**	0.9863
	2.	**0.9915**	0.9874
	3.	**0.9929**	0.9865
	4.	**0.9917**	0.9857
	5.	**0.9917**	0.9866
	6.	**0.9924**	0.9870
	7.	**0.9905**	0.9870
	8.	**0.9913**	0.9878
	9.	**0.9925**	0.9856
	10.	**0.9923**	0.9841
	$\overline{F1}$	**0.9919**	0.9864
	σ	**0.0007**	0.0011
	F_{med}	**0.9919**	0.9866

Tablica 5.24: *Usporedba performansi PSO i AutoKeras algoritama na Fashion MNIST skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO	AutoKeras
Fashion MNIST	1.	**0.9207**	0.9065
	2.	**0.9160**	0.9069
	3.	**0.9163**	0.9031
	4.	**0.9200**	0.9027
	5.	**0.9198**	0.8998
	6.	**0.9186**	0.9060
	7.	**0.9203**	0.9027
	8.	**0.9172**	0.9065
	9.	**0.9175**	0.9037
	10.	**0.9163**	0.9063
	$\overline{F1}$	**0.9183**	0.9044
	σ	**0.0018**	0.0024
	F_{med}	**0.9180**	0.9049

Tablica 5.25: *Usporedba performansi PSO i AutoKeras algoritama na CIFAR-10 skupu podataka*

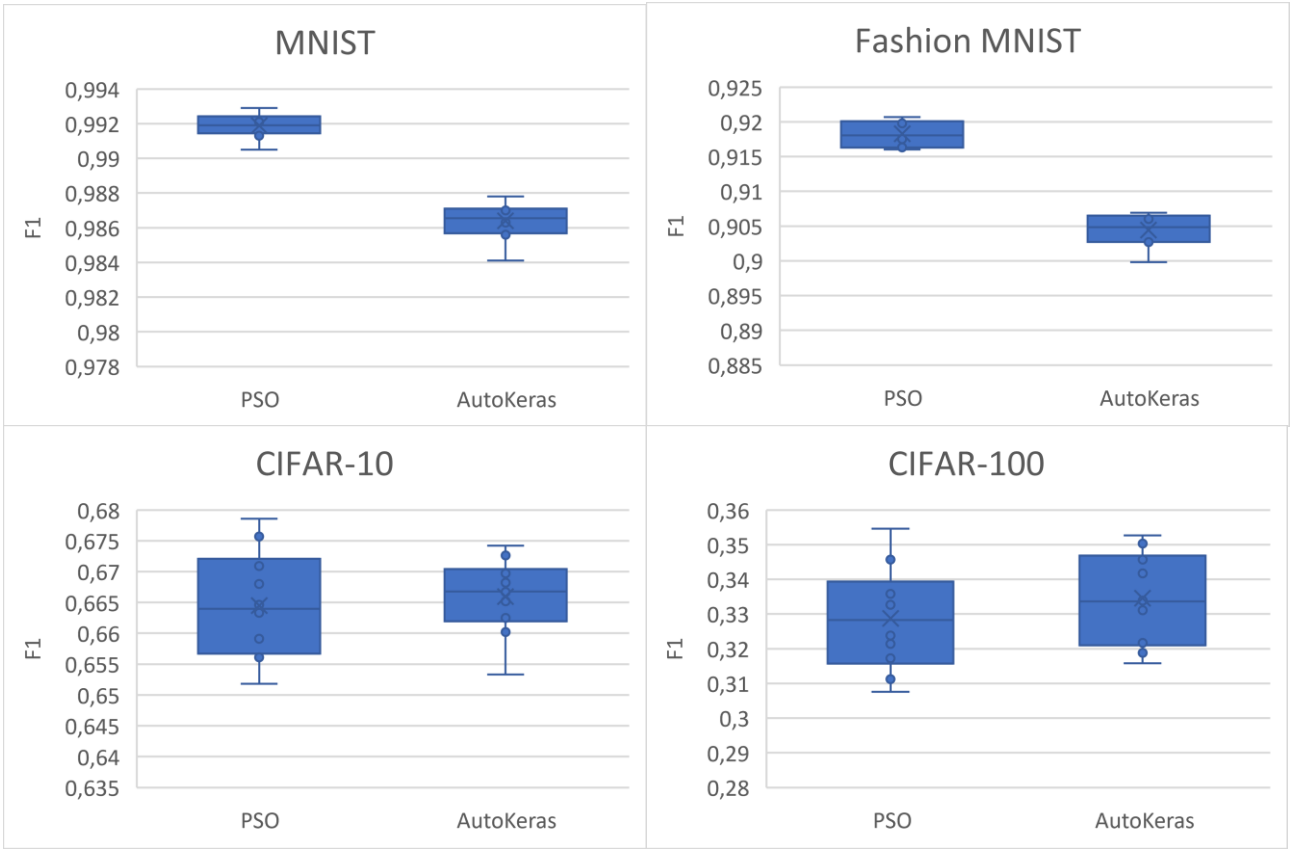
Skup podataka	Izvođenje eksperimenta	PSO	AutoKeras
CIFAR - 10	1.	**0.6569**	0.6669
	2.	**0.6709**	0.6533
	3.	0.6591	**0.6697**
	4.	0.6633	**0.6652**
	5.	**0.6680**	0.6625
	6.	0.6561	**0.6667**
	7.	0.6518	**0.6602**
	8.	0.6646	**0.6726**
	9.	**0.6757**	0.6742
	10.	**0.6786**	0.6682
	$\overline{F1}$	0.6645	**0.6659**
	σ	0.0088	**0.0061**
	F_{med}	0.6640	**0.6668**

Tablica 5.26: *Usporedba performansi PSO i AutoKeras algoritama na CIFAR-100 skupu podataka*

Skup podataka	Izvođenje eksperimenta	PSO	AutoKeras
CIFAR - 100	1.	**0.3327**	0.3158
	2.	0.3112	**0.3457**
	3.	0.3076	**0.3217**
	4.	**0.3358**	0.3188
	5.	**0.3546**	0.3417
	6.	**0.3457**	0.3334
	7.	0.3373	**0.3503**
	8.	0.3238	**0.3311**
	9.	0.3172	**0.3339**
	10.	0.3214	**0.3527**
	$\overline{F1}$	0.3287	**0.3345**
	σ	0.0151	**0.0130**
	F_{med}	0.3283	**0.3336**

Tablica 5.27: *Sažete performanse algoritama PSO i AutoKeras*

Skup podataka	PSO		AutoKeras	
	Broj najboljih rezultata	$\overline{F1}$	Broj najboljih rezultata	$\overline{F1}$
MNIST	**10**	**0.9919**	0	0.9864
Fashion MNIST	**10**	**0.9183**	0	0.9044
CIFAR-10	5	0.6645	5	**0.6659**
CIFAR-100	4	0.3287	**6**	**0.3345**



Slika 5.5: *Usporedba performansi najboljih mreža pronađenih algoritama PSO i AutoKeras*

6. ZAKLJUČAK

Automatski dizajn neuronskih mreža nije jednostavan problem te postoji mnogo postupaka za njegovo rješavanje. Ovaj problem može se predstaviti kao optimizacijski problem zbog toga što je potrebno podesiti strukturu mreže definiranjem niza parametara širokog raspona mogućih vrijednosti. Dakako, prikladne vrijednosti ovih parametara uvelike ovise o karakteristikama problema za koji se mreža primjenjuje. U literaturi su primijenjene brojne metode optimizacije, među kojima se ističu bio-inspirirani algoritmi zbog svoje učinkovitosti i sposobnosti usmjerenog pretraživanja pretrage. Algoritam PSO jedan je od predstavnika ovih algoritama koji je demonstrirao svoju učinkovitost u brojnim različitim problemima optimizacije, pa tako i u rješavanju problema automatskog dizajna neuronskih mreža. Algoritam PSO sadrži razne parametre koji utječu na njegovo ponašanje, a time i na njegovu učinkovitost. Uz to, potrebno mu je odrediti prikladan način predstavljanja rješenja za dani problem. U prvom dijelu eksperimentalne analize uspoređena su dva načina predstavljanja rješenja. Utvrđeno je kako predstavljanje rješenja vektorima realnih brojeva rezultira boljim performansama nego predstavljanje rješenja vektorima binarnih brojeva. U drugom dijelu eksperimentalne analize utvrđeno je da generalno najbolje performanse daju postavke algoritma PSO-6 ($\omega = 0.724$, $c_1 = 1.468$, $c_2 = 1.468$) preuzete iz literature [22], koje izjednačavaju utjecaj globalne i osobne komponente svake čestice. U sljedećem dijelu eksperimentalne analize uspoređen je algoritam PSO s nekim od poznatijih i jednostavnijih načina za podešavanje hiperparametara algoritama strojnog učenja, koji se također često koriste i za dizajniranje neuronskih mreža. Kako bi se postigao općenitiji uvid u performanse algoritma, eksperimentalna analiza provedena je na četiri različita skupa podataka. Eksperimentalnom analizom utvrđeno je da algoritam PSO ima bolje performanse u odnosu na klasične algoritme. Povrh toga, usporedbom s AutoKeras algoritmom koji predstavlja gotovo rješenje za automatski dizajn neuronskih mreža utvrđeno je da algoritam PSO generalno ima bolje performanse.

Eksperimentalnom analizom algoritam PSO pokazao se kao dobra metoda optimizacije za automatski dizajn neuronskih mreža. Eksperimentalna analiza provodila se nad manjim skupom hiperparametara te uvođenje novih hiperparametara i proširivanje njihovih raspona mogu biti smjernice za budući rad. Osim hiperparametara, ovim radom pokazalo se kako način predstavljanja rješenja utječe na performanse algoritma. U eksperimentu koristila su se dva načina predstavljanja rješenja od kojih se predstavljanje rješenja kao vektor realnih brojeva pokazao boljim. Međutim, moguće je da neki drugi načini predstavljanja rješenja opisani u literaturi mogu imati bolje

performanse. Usporedba s drugim bio-inspiriranim algoritmima može dati bolji uvid u performanse algoritma PSO. Osim primjene bio-inspiriranih algoritama, u literaturi se navode brojni drugi postupci za automatski dizajn neuronskih mreže. Usporedba s drugim postupcima koji provode Bayesovu optimizaciju ili koriste podržano učenje može biti dio budućeg istraživanja.

Literatura

[1] I. Goodfellow, Y. Bengio i A. Courville. Deep Learning. MIT Press, str. 326-367, 2016.

[2] A. P. Engelbrecht. Computational Intelligence: An Introduction, Second Edition. 2002.

[3] M. Dudjak. Učenje iz neuravnoteženih podataka unaprijeđenim postupcima za odabir značajki, preuzorkovanje i izgradnju radijalnih neuronski mreža. Sveučilište Josipa Jurja Strossmayera, FERIT, Osijek, 2022.

[4] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 1980

[5] D. H. Hubel i T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. J Physiol, 1968.

[6] S. Khan, H. Rahmani i S. A. A. Shah. A Guide to Convolutional Neural Networks for Computer Vision. Morgan & Claypool Publishers, 2018.

[7] G. Zaccone, R. Karim i A. Menshawy. Deep Learning with TensorFlow. 2017.

[8] M. Wistuba, A. Rawat i T. Pedapati. A Survey on Neural Architecture Search. 2019. pristipljeno: 25.06.2023., http://arxiv.org/abs/1905.01392

[9] G. Kyriakides i K. Margaritis. An Introduction to Neural Architecture Search for Convolutional Networks. 2020, pristupljeno: 25.06.2023., http://arxiv.org/abs/2005.11074

[10] P. Ren, Y. Xiao, X. Chang, P. Y. Huang, Z. Li, X. Chen i X. Wang. A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. 2021., pristupljeno: 25.06.2023., http://arxiv.org/abs/2006.02903

[11] T. Elsken, J. H. Metzen i F. Hutter. Neural Architecture Search: A Survey. 2019.

[12] S. Garg, K. Patra i S. K. Pal. Particle Swarm Optimization of a Neural Network Model in a Machining Process. Sadhana, 2014.

[13] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le i A. Kurakin. Large-scale evolution of image classifiers. 2017.

[14] H. Pham, M. Guan, B. Zoph, Q. Le i J. Dean. Efficient Neural Architecture Search via Parameters Sharing. 2018.

[15] B. Wang, Y. Sun, B. Xue i M. Zhang. Evolving Deep Convolutional Neural Networks by Variable-length Particle Swarm Optimization for Image Classification. 2018. pristupljeno: 25.06.2023, http://arxiv.org/abs/1803.06492

[16] J. Kennedy i R. Eberhart. Particle swarm optimization. PurdueSchool of Engineering and Technology, Indianapolis, 1995.

[17] H. P. Singh i A. Kaur. Statistical analysis of velocity update rules in particle swarm optimization. Guru Nanak Dev University, Amritsar, Computer Science Department, India, 2015.

[18] B. Zorić, D. Bajer i M. Dudjak. Wrapper-based feature selection via differential evolution: benchmarking different discretisation techniques. Osijek, 2020.

[19] „Keras documentation“ https://keras.io/api/, Zadnje pristupljeno: 05.12.2023.

[20] „Skimage documentation”, https://scikit-image.org/, Zadnje pristupljeno: 10.12.2023.

[21] A. Krizhevsky, I. Sutskever i G. E. Hinton. Imagenet classification with deep convolutional neural networks. Curran Associates, Inc., str. 1097–1105., 2012.

[22] K. Harrison, B. Ombuki-Berman i A. Engelbrecht. A Parameter-Free Particle Swarm Optimization Algorithm using Performance Classifiers. 2019.

[23] H. Jin, F. Chollet, Q. Song i X. Hu. AutoKeras: An AutoML Library for Deep Learning. Journal of machine Learning research 6: 1-6, 2023.

[24] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen i K. C. Tan. A Survey on Evolutionary Neural Architecture Search. IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 2, str. 550–570, 2023, 10.1109/TNNLS.2021.3100554.

[25] P. G. Devarakonda i B. Bozic. Particle Swarm Optimization of Convolutional Neural Networks for Human Activity Prediction. IntechOpen, 2022., 10.5772/intechopen.97259

[26] B. Xue, M. Zhang i W. N. Browne. Particle Swarm Optimization for Feature Selection in Classification: A Multi-Objective Approach. 2013.

[27] D. Anghinolfi i M. Paolucci. A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. European Journal of Operational Research, Volume 193, Issue 1, str 73-85, ISSN 0377-2217, 2009. https://doi.org/10.1016/j.ejor.2007.10.044.

[28] E. C. Laskari, K. E. Parsopoulos i M. N. Vrahatis. Particle swarm optimization for integer programming. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, Honolulu, HI, USA, 2002, str. 1582-1587 vol.2, doi: 10.1109/CEC.2002.1004478.

[29] M. Carvalho i T. B. Ludermir. Particle Swarm Optimization of Neural Network Architectures andWeights. *7th International Conference on Hybrid Intelligent Systems (HIS 2007)*, Kaiserslautern, Germany, 2007, str. 336-339, doi: 10.1109/HIS.2007.45.

Popis oznaka i kratica

CNN	Konvolucijska neuronska mreža
PSO	Optimizacija rojem čestica
ANN	Umjetna neuronska mreža
NAS	Pretraživanje neuronske arhitekture
RL	Podržano učenje
ES	Evolucijska strategija
GA	Genetski algoritam
p_{best}	Najbolji osobni položaj čestice
g_{best}	Najbolji globalni položaj cijelog roja
v	Brzina čestice
ω	Inercijska težina
c_1	Kognitivni akceleracijski koeficijent
c_2	Socijalni akceleracijski koeficijent
x	Trenutni položaj čestice
N_s	Broj čestica u populaciji
TP	Broj ispravnih pozitivnih rezultata
FN	Broj neispravnih negativnih rezultata
FP	Broj neispravnih pozitivnih rezultata
σ	Standardna devijacija
$\overline{F1}$	Prosječna vrijednost mjere F1
$F1_{med}$	Medijalna vrijednost mjere F1
RS	Algoritam nasumičnog pretraživanja
ES	Algoritam iscrpnog pretraživanja
RGS	Algoritam nasumičnog i mrežnog pretraživanja
$\overline{N}$	Prosječna vrijednost hiperparametra

Popis slika

Popis tablica

Sažetak

Konvolucijske neuronske mreže primjenjuju se za rješavanje raznih problema dubokog učenja. Primjena konvolucijskih neuronskih mreža za zadani problem zahtijeva dizajniranje odgovarajuće strukture mreže. Ručni odabir i vrednovanje potencijalnih struktura mreža zamoran je i dugotrajan proces. Automatizacija ovog postupka uvelike može smanjiti napore i vrijeme potrebno za dizajniranje prikladnih mreža. Automatski dizajn neuronskih mreža može se predstaviti kao optimizacijski problem te je u literaturi predloženo pregršt postupaka za njegovo rješavanje. U ovom radu korišten je algoritam PSO koji se pokazao učinkovitim u rješavanju brojnih problema optimizacije u literaturi, među kojima je i dizajn neuronskih mreža. Kako bi se performanse algoritma PSO mogle usporediti s nekom od poznatijih i jednostavnijih algoritama za dizajniranje neuronskih mreža, proveden je postupak pronalaska prikladne inačice algoritma PSO u smislu načina predstavljanja rješenja i odabira vrijednosti njegovih parametara. Eksperimentalna analiza provedena je na četiri skupa podataka: MNIST, Fashion MNIST, CIFAR-10 i CIFAR-100. Eksperimentalnom analizom utvrđeno je kako je najbolji način predstavljanja rješenja pomoću vektora realnih vrijednosti, a kod odabira parametara povoljnim se pokazalo odabrati jednake vrijednosti akceleracijskih koeficijenata. Eksperimentalnom analizom također je utvrđeno kako algoritam PSO postiže bolje rezultate od uobičajeno korištenih postupaka za dizajniranje neuronskih mreža, što ga čini prikladnim algoritmom za ovaj problem.

Ključne riječi: duboko učenje, konvolucijske neuronske mreže, optimizacija rojem čestica, pretraživanje neuronske arhitekture

Abstract

Convolutional neural networks are applied to solve various deep learning problems. Applying convolutional neural networks to a given problem requires designing an appropriate network structure. Manual selection and evaluation of potential network structures is a tedious and time-consuming process. Automating this process can significantly reduce the effort and time needed to design suitable networks. The automatic design of neural networks can be presented as an optimization problem, and a handful of procedures for solving it have been proposed in the literature. In this paper, the PSO algorithm was used, which proved to be effective in solving numerous optimization problems in the literature, among which is the design of neural networks. To compare the performance of the PSO algorithm with more well-known and simpler algorithms for neural network design, a process of finding a suitable version of the PSO algorithm in terms of solution representation and parameter value selection was conducted. Experimental analysis was carried out on four datasets: MNIST, Fashion MNIST, CIFAR-10 and CIFAR-100. Through experimental analysis, it was determined that the best way to represent solutions is through a vector of real values and selecting equal values for acceleration coefficients proved favorable in parameter selection. The experimental analysis also revealed that the PSO algorithm achieves better results than commonly used procedures for neural network design, making it a suitable algorithm for this problem.

Keywords: deep learning, convolutional neural networks, particle swarm optimization, neural arhitecture search

Životopis

Ante Ukić rođen je u Vinkovcima 19.10.1999. godine. U Otoku je pohađao osnovnu školu Josipa Lovretića. Nakon završene osnovne škole 2014. godine upisuje smjer tehničke gimnazije u Tehničkoj školi Ruđera Boškovića u Vinkovcima. Nakon završetka srednje škole upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku kojeg završava 2021. godine, a 2021. godine na istom fakultetu upisuje diplomski studij računarstva. Od svibnja 2022. godine zaposlen je na poziciji backend developera u Financijskoj Agenciji.