GRADNJA KONVOLUCIJSKIH NEVRONSKIH MREŽ ZA RAZVRŠČANJE SLIK Z UPORABO EVOLUCIJSKIH ALGORITMOV

UROŠ MLAKAR
Univerza v Mariboru, Fakulteta za elektrotehniko računalništvo in informatiko, Maribor, Slovenija
uros.mlakar@um.si

Globoke nevronske mreže so bile uspešno aplicirane že na mnogih področjih obdelave slik. Uspešnost mreže je vedno pogojena z njeno arhitekturo, ki pogosto zahteva ročno oblikovanje strokovnjaka z bogatim strokovnim znanjem. Takšen pristop je lahko v realnem svetu zamuden, morda tudi neizvedljiv, predvsem zaradi primanjkljaja izkušenj načrtovalcev oz. njihovega znanja. V tem članku smo predstavili postopek avtomatskega iskanja topologije globoke nevronske mreže v aplikaciji razvrščanja slik. Iskanje ustrezne topologije smo preslikali v optimizacijski problem, ki ga rešujemo z algoritmom diferencialne evolucije. Algoritem smo testirali nad podatkovnima zbirkama CIFAR10 in AffectNet. Dobljeni rezultati so obetavni in odpirajo novo mlado razsikovalno področje načrtovanja globokih mrež brez predhodnega ekspertnega znanja.

DOI
https://doi.org/10.18690/um.feri.1.2024.3

ISBN
978-961-286-837-6

Ključne besede:
evolucijski algoritem, globoka mreže, razvrščanje slik, generiranje globokih mrež, optimizacija

Univerzitetna založba Univerze v Mariboru

DOI
https://doi.org/
10.18690/um.feri.1.2024.3

ISBN
978-961-286-837-6

Building Convolutional Neural Networks for Image Classification Using Evolutionary Algorithms

Uroš Mlakar
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia
uros.mlakar@um.si

Keywords:
evolutionary algorithm, deep networks, image classification, generating deep networks, optimization

Deep neural networks have already been successfully applied in many areas of image processing. The success of a network is always conditioned by its architecture, which requires manual design with expert knowledge. In the real world, such an approach can be time-consuming, perhaps even unfeasible, mainly due to the lack of experience of users or of their knowledge. In this paper, we presented the process of automatically finding the topology of a deep network in an image classification application. We mapped the search for a suitable topology into an optimization problem, which is solved using the differential evolution algorithm. We tested the algorithm on the CIFAR10 and AffectNet datasets. The obtained results are promising and open up a new young research field of designing deep networks without prior expert knowledge.

University of Maribor Press

1 Uvod

Globoke nevronske mreže (angl. Deep Neural Networks - DNN), ki so temelj globokega učenja (cite), so se v zadnjih letih izkazale uspešne pri različnem naboru aplikacij, kjer so največ uporabljene pri razvrščanju slik (He, 2016) (Huang, 2017), procesiranju naravnega jezika, prepoznavanju govora (Zhang, 2017) in drugih. Njihova pomembna lastnost je sposobnost prepoznavanja in učenja pomembnih lastnosti vhodnih podatkov, brez predhodnega ročnega oblikovanja značilnic. V splošnem uspešnost DNN pogojujeta dva vidika in sicer zasnova arhitekture ter pripadajoče uteži. Šele ko sta oba hkrati v optimalnem stanju, lahko pričakujemo obetavno delovanje DNN. Optimalne uteži se pogosto pridobijo s postopkom učenja: z uporabo zvezne funkcije izgube za merjenje razlik med dejanskim in želenim izhodom, nato pa se za minimiziranje izgube pogosto uporabljajo algoritmi, ki temeljijo na gradientnem spustu. Ko je izpolnjen pogoj za zaključek učenja, ki je običajno število učnih iteracij, lahko algoritem pogosto najde dober nabor uteži (LeCun, 2015). Takšen postopek je v praksi zelo priljubljen predvsem zaradi učinkovitosti in se ga največ uporablja za optimizacijo uteži, čeprav gre zgolj za algoritme lokalnega iskanja. Po drugi strani pa iskanje optimalnih arhitektur ni mogoče neposredno formulirati z zvezno funkcijo, ne obstaja pa tudi eksplicitna funkcija za merjenje procesa iskanja optimalne arhitekture.

Obetavne arhitekture, ki dosegajo dobre rezultate večinoma razvijejo strokovnjaki z obširnim domenskim strokovnim znanjem. Primer takšnih arhitektur so npr. VGG (Zisserman, 2015), ResNet (Sun, 2015) in DenseNet (Weinberger, 2018). Vse omenjene modele konvolucijskih nevronskih mrež (angl. Convolutional Neural Networks - CNN) so ročno zasnovali raziskovalci z bogatim znanjem s področja nevronskih mrež in obdelave slik. Pogosto se dogaja, da v praksi večina končnih uporabnikov nima takšnega znanja. Poleg tega so arhitekture DNN/CNN pogosto odvisne tudi od reševanega problema. Če se porazdelitev vhodnih podatkov spremeni, je treba arhitekturo ustrezno spremeniti. Proces iskanja arhitektur globokih mrež (angl. Neural Architecture Search - NAS), katerega cilj je avtomatizacija načrtovanja arhitekture globokih nevronskih mrež, je opredeljen kot obetaven način za reševanje zgoraj navedenih izzivov. Matematično lahko NAS modeliramo kot optimizacijski problem:

$$argmin_{(A \in \Lambda)} = \Gamma\left(A, D_{učna}, D_{uspešnost}\right) \quad (1)$$

V enačbi (1) je Λ iskalni prostor vseh potencialnih arhitektur, $\Gamma(\ )$ pa meri uspešnost posamezne arhitekture, ki je naučena z učno množico D_{train}.

Načeloma je NAS težek optimizacijski problem, postavljen pred več izzivov, kot so kompleksne omejitve, diskretne predstavitve, dvonivojske strukture, računske zahtevnosti in številna nasprotujoča si merila. Algoritmi NAS se nanašajo na optimizacijske algoritme, ki so posebej zasnovani za učinkovito in uspešno reševanje problema, ki ga predstavlja enačba (1).

NAS algoritme lahko glede na uporabljen optimizacijski algoritem delimo v 3 kategorije:

- Algoritmi, ki temeljijo na učenju z okrepitvijo (angl. Reinforcement Learning -LR) (Kaelbling, 1996),
- Algoritmi, ki temeljijo na gradientu in
- Evolucijski algoritmi (Back, 1997).

Algoritmi, ki uporabljajo RL, pogosto zahtevajo ogromno grafičnih kartic že za preprostejše probleme razvrščanja slik. Algoritmi, ki temeljijo na gradientu, so sicer bolj učinkoviti od algoritmov, ki temeljijo na RL, vendar pogosto najdejo slabše arhitekture zaradi neustreznega razmerja skaliranja za optimizacijski algoritem. Dodatno je pri načrtovanju teh algoritmov potrebno že precej ekspertnega znanja za učinkovito reševanje problema. V tretjo skupino spadajo pa evolucijski algoritmi, ki temeljijo na populaciji rešitev in simulirajo razvoj vrste ali obnašanje populacije v naravi. Sposobni so poiskati rešitev problema tudi kadar matematična oblika kriterijske funkcije ni na voljo. Ta skupina algoritmov ja za problem NAS sicer zelo zanimiva, vendar je podobno kot pristop z RL precej računsko požrešna.

V zadnjih nekaj letih se v literaturi pojavlja kar nekaj pristopov načrtovanja arhitektur globokih mrež z uporabo evolucijskih algoritmov na različnih področjih, kar nakazuje na očitno pomanjkanje ekspertnega znanja. V tem članku bomo predstavili algoritem diferencialne evolucije (angl. Differential Evolution - DE), ki spada v skupino evolucijskih algoritmov. Uporabili ga bomo za iskanje optimalne arhitekture CNN za izbran problem razvrščanja slik. Pri tem ne bomo uporabljali domenskega ekspertnega znanja.

2 Diferencialna evolucija

Diferencialna evolucija (DE) je stohastični populacijski algoritem, ki se uporablja za reševanje zveznih in tudi diskretnih problemov (Mlakar, 2017) (Brest, 2006). Navkljub svoji enostavnosti je izredno učinkovit pri reševanju realnih problemov. Temelji na matematičnem modelu, ki uporablja razlike vektorjev. DE algoritem razvija in spreminja populacijo vektorjev skozi generacije, pri čemer gre vsak vektor skozi niz evolucijskih operatorjev, vključno z mutacijo, križanjem in selekcijo. Populacijo v okviru algoritma DE lahko matematično zapišemo kot (Mlakar U. , 2019):

$$x_i^{(g)} = \left(x_{(i,1)}^{(g)}, \dots, x_{(i,D)}^{(g)}\right), \text{za } i = 1, \dots, Np, \tag{2}$$

kjer je D dimenzija reševanega problema, Np je pa velikost populacije. Algoritem DE v prvem koraku izvede operator mutacije, nato pa mutiran vektor križa. Predlaganih je bilo že precej različnih mutacij, vendar se v algoritmu DE najpogosteje še vedno uporablja metoda "rand/1", ki jo matematično definiramo kot (Mlakar U. , 2019):

$$v_i^{(g)} = x_{r_1}^{(g)} + F\left(x_{r_2}^{(g)} - x_{r_3}^{(g)}\right). \tag{3}$$

V enačbi (3) so r_1, r_2 in r_3 naključno generirane celoštevilske vrednosti na intervalu

$[1, Np]$, F je skalirni faktor na intervalu $[0,1]$, $v_i^{(g)}$ pa je mutiran vektor. Dobljen mutiran vektor $v_i^{(g)}$ je v drugem koraku križan z originalnim vektorjem $x_i^{(g)}$, kjer kot rezultat dobimo poskusni vektor. Križanje matematicno zapišemo kot (Mlakar U. , 2019):

$$u_{i,j}^{(g)} = \begin{cases} v_{i,j}^{(g)}, \check{c}e rand(0,1) \leq Cr \text{ ali } j = j_{rand}, \\ x_{i,j}^{(g)}, sicer. \end{cases} \tag{4}$$

V enačbi (4) je verjetnost križanja Cr definirana na intervalu $[0,1]$. Dodatni pogoj $j = j_{rand}$, kjer je $j_{rand} = rand(0, D)$, zagotavlja, da bo poskusni vektor $u_i^{(g)}$ od originalnega različen vsaj v enem elementu. S tem preprečimo izdelavo enakih posameznikov in posledično ne izgubljamo časa z ocenjevanjem rešitve, ki smo jo že ocenili. V zadnjem koraku evolucijskega procesa poskusni vektor tekmuje z originalnim v postopku selekcije. Vektor, katerega funkcija uspešnosti je boljša, je izbran za preživetje v naslednjo generacijo. Matematično operator selekcije zapišemo kot (Mlakar U. , 2019):

$$x_i^{(g+1)} = \begin{cases} u_i^{(g)}, \check{c}e f\left(u_i^{(g)}\right) \leq f\left(x_i^{(g)}\right), \\ x_i^{(g)}, sicer. \end{cases} \quad (4)$$

3 Algoritem diferencialne evolucije za reševanje problema NAS

Največji izziv pri uporabi evolucijskih algoritmov za reševanje problem kot je NAS, je učinkovita in smiselna predstavitev posameznika v populaciji. V literaturi so pojavljata predvsem dva načina, in sicer predstavitev na podlagi plasti, ter predstavitev na podlagi blokov. Pri prvem so osnovni elementi v kodirnem prostoru kodiranja osnovne plasti, kot so npr. kovolucijske ali polno povezane. Pri tem načinu se pojavlja težava ogromnega iskalnega prostora, saj poskušamo zakodirati veliko količino informacij. Skoraj zagotovo bom s tem kodiranjem porabili tudi več časa, da najdemo primerno rešitev, predvsem zato, ker je težje sestaviti dobro delujočo DNN/CNN zgolj z osnovnimi sloji. S takim kodiranjem zagotovo ne moremo poiskati oz. najti mreže, ki je po topologiji podobna Resnet-u. Pri drugem načinu kodiranja, pa so osnovni gradniki v iskalnem prostoru bloki, ki so se že izkazali kot primerni v drugih tipih mrež. S tem načinom kodiranja sicer lahko pohitrimo postopek iskanja optimalne mreže, vendar potrebujemo že nekaj ekspertnega znanja, saj moramo izbrati takšne bloke, ki bodo ustrezali reševanemu problemu. V tem delu smo izbrali srednjo pot, kjer smo kombinirali obe predstavitvi.

Vsaka rešitev v populaciji algoritma DE predstavlja globoko arhitekturo, ki jo je potrebno ustrezno preslikati iz iskalnega prostora v prostor rešitev. To preslikavo naredimo z ustreznimi preslikovalnimi funkcijami, s katerimi zagotovimo pravilnost zapisa arhitekture in jo s tem pripravimo na učenje. Delovanje preslikovalnih funkcij je determinstično, saj se določena vrednost v določenih mejah iz iskalnega prostora

vedno preslika v enak sloj. Vsak sloj nosi s seboj v prostoru rešitev tudi dodatne informacije, ki so potrebne za pravilno preslikavo. Globina arhitekture mreže je odvisna od dimezije rešitve v iskalnem prostoru.

Vsaka dimenzija torej predstavlja gradnik mreže, ki je lahko osnovni sloj ali pa sestavljen blok. V tem delu smo uporabili naslednje osnovne sloje:

- Konvolucijski sloj,
- Polno povezan sloj (angl. Fully-Connected layer) in
- Združevalni sloj (angl. Pooling layer),
- Izpustni sloj (angl. Dropout layer) ter

naslednje bloke:

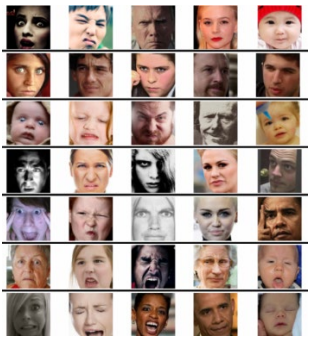
- Rezidualni blok (angl. Resnet Block), kot je uporabljen v ResNet arhitekturi in
- Gost blok (angl. Dense Block), kot je uporabljen v DenseNet arhitekturi.

Konvolucijski sloj ima v prostoru rešitev dodano število uporabljenih filtrov, polno povezan sloj nosi informacije o številu nevronov ki jih vsebuje, združevalni sloj pa definira še tip združevanja (maksimalno ali povprečno). Izpustni sloj nosi informacije o odstotku povezav, ki jih bo ignoriral. Oba uporabljena bloka sta zaradi manjšanja velikosti iskalnega prostora in posledično računske zahtevnosti nespremenljiva.

4 Eksperimentalno okolje

V tem poglavju bomo opisali eksperimentalno okolje, v katerem so tekli vsi eksperimenti v okviru tega dela. Za testiranje predlaganega algoritma DE za iskanje optimalnih globokih arhitektur, smo uporabili podatkovni zbirki CIFAR10 (Hinton, 2009) in AffectNet (Mollahosseini, 2019). Podatkovna zbirka CIFAR10 se pogosto pojavlja pri validaciji rezultatov globokih mrež, ne glede na izbran pristop gradnje mreže. Vsebuje 60000 slik, ki jih je možno razvrstiti v 10 razredov. Vsaka slika je velikosti 32 x 32 slikovnih elementov v barvnem prostoru RGB. Na drugi strani pa podatkovna zbirka Affectnet vsebuje približno 440000 slik, ki vsebuje slike obraznih

izrazov, ki jih je možno razvrstiti v 8 razredov. V tej podatkovni zbirki so slike velike 96 x 96 slikovnih elementov, tudi v barvnem prostoru RGB.



Slika 1: Vzorci slik iz podatkovne zbirke AffectNet.
Vir: lasten

Za potrebe validacije predlaganega algoritma na problemu NAS smo podatkovno zbirko razdeliti na učno in testno množico. CIFAR10 ima sicer ti množici že strogo definirani, kjer se v učni množici nahaja 50000 slik, v testni pa 10000. Pri podatkovni zbirki AffectNet smo delitev morali narediti sami. Razdelitev smo opravili v razmerju 90 % za učno množico in 10 % za testno. Razdelitev smo naredili na nivoju števila obraznih izrazov. Dodatno smo pri delitvi zagotovili, da se slike, ki so bile izbrane za učno množico, niso pojavile hkrati še v testni. Za potrebe učenja in zagotavljanja generalizacije naučene mreže, smo dodatno iz učne množice izvzeli 20 % podatkov, ki smo jih uporabili za validacijsko množico. Ta se je uporabljala za ovrednotenje kriterijske funkcije, ko je arhitektura končala z učenjem na učni množici po določenem številu epoh. Ko je algoritem zaključil z iskanjem, smo najboljšo najdeno arhitekturo testirali na testni množici. Uspešnost razvrščanja nad testno množico je tudi končni rezultat najdene mreže.

Parametre algoritma, kot so velikost populacije, število generacij, začetna dimenzija problema, število epoh in velikost paketa smo nastavljali eksperimentalno. Hkrati smo spreminjali tudi dovoljene gradnike v arhitekturah. Za učni algoritem smo izbrali algoritem Adam s privzetimi nastavitvami v knjižnici Keras. Ker gre za računsko težek problem, smo algoritem, napisan v programskem jeziku python, prilagodili za izvajanje na superračunalniku Vega, kjer smo poskrbeli za sočasno ovrednotenje celotne populacije na trenutno razpoložljivih virih.

5 Rezultati

V tem poglavju bomo predstavili rezultate eksperimentalnega dela. Cilj eskperimentov je bil pokazati, da je možno z algoritmom DE najti smiselne arhitekture za izbran problem razvrščanja slik, brez posredovanja ekspertnega znanja iskalnemu algoritmu. Dodatno smo želeli še preveriti, kako velikost učne množice vpliva na uspešnost iskanja optimalne arhitekture. Ker spada algoritem DE med stohastične algoritme, smo ga za vsako kombinacijo parametrov zagnali 5 krat. V Tabeli 1 so zbrani rezultati za podatkovno zbirko CIFAR10, v Tabeli 2 pa rezultati za AffectNet. Stolpec gradniki v obeh tabelah opisuje dovoljene sloje in bloke v zagonu algoritma. V zadnjih dveh stolpcih je zapisana povprečna napaka razvrščanja in standardni odklon petih zagonov nad validacijsko in testno množico. Naj še poudarimo, da se je algoritem pri določenih kombinacijah parametrov ujel v lokalni optimum, zato teh rezultatov ne poročamo.

Tabela 1: Rezultati algoritma DE nad podatkovno zbirko CIFAR10.

Gradniki[1]	Vel. Pop.	Gen.	% učne množice	Dim	#Epoh	Vel. paketa	Nap. na val.	Nap. na test.
P,C,S,D,F,B	20	20	0,4	30	100	128	20,99 (3,3)	7,44 (2,3)
P,C,S,D,F,B	20	20	0,4	40	100	128	24,75 (2,4)	9,56 (1,4)
P,C,D,F,B	20	20	0,4	40	50	64	19,99 (3,6)	10,53 (2,7)
P,C,S,D,F,B	20	20	0,4	30	100	128	21,79 (5,5)	9,35 (3,3)
P,C,D,F,B	20	20	0,8	40	100	128	15,2 (4,3)	9,04 (2,6)
P,C,D,F,B	20	20	0,8	50	100	128	17,4 (6,1)	10,79 (1,1)
P,C,D,F,B	20	20	0,8	60	100	128	15,32 (2,1)	9,46 (0,8)
P,C,D,F,B	20	20	0,8	70	100	128	17,44 (3,8)	11,61 (2,2)
P,C,S,D,F,B	20	20	0,8	20	100	128	19,8 (3,9)	9,64 (1,6)
P,C,S,D,F,B	20	20	1	30	10	128	38,59(5,8)	14,63 (3,4)
P,S,F,B	**20**	**30**	**1**	**10**	**50**	**64**	**12,44 (1,6)**	**8,3 (0,4)**
P,S,F,B	20	40	1	10	10	128	25,9 (3,1)	10,57 (2,2)
P,S,F,B	20	40	1	10	10	64	20,93 (3,4)	9,29 (1,1)
P,S,F,B	20	40	1	15	10	128	27,34 (1,3)	12,62 (2,2)
P,S,F,B	20	40	1	15	30	128	16,25 (4,3)	10,25 (2,5)
P,S,F,B	20	40	1	15	30	256	24,27 (3,9)	11,94 (3,1)

[1] P-Združevalni sloj; C-Konvolucijski sloj; F-Polno povezan sloj; D-Izpustni sloj; S-Rezidualni blok; B-Gost blok

Tabela 2: Rezultati algoritma DE nad podatkovno zbirko AffectNet.

Gradniki	Vel. Pop.	Gen.	% učne množice	Dim	#Epoh	Vel. paketa	Nap. na val.	Nap. na test.
P,S,F,C,D	20	20	0,1	10	20	128	33,02 (2,4)	52,35 (3,3)
P,S,F,B	20	20	0,18	5	10	256	34,10 (3,8)	56,68 (4,1)
P,S,F,B	**20**	**20**	**0,19**	**10**	**10**	**128**	**29,18 (5,2)**	**50,55 (2,4)**

Iz Tabele 1 lahko razberemo, da dobimo najboljši rezultat, ko se uporabljamo pretežno enostavne sloje in celotno podatkovno zbirko v procesu učenja. Tukaj smo dosegli najnižjo napako pri 8,3 %. V tem primeru je imel algoritem tudi več časa za samo optimizacijo s 600 ovrednotenji kriterijske funkcije, kar pa sicer v splošnem ne zagotavlja boljših rezultatov. Ugotovili smo tudi, da velikost učne množice nima večjega vpliva na rezultate, saj so napake razvrščanja pri vseh variantah velikosti zelo podobne (< 2 %), se pa računska zahtevnost zagotovo zmanjša. V literaturi najboljše metode iz družin evolucijskih algoritmov na podatkovni zbirki CIFAR10 dosegajo napake manjše tudi od 5 %, kjer pa z določenimi omejitvami algoritma manjšajo iskalni prostor. S tem seveda tudi vnašajo ekspertno znanje v reševanje problema, čemu smo mi skušali izogniti.

Na podatkovni zbirki AffectNet smo dosegli najnižjo napako razpoznave pri 50,55 %, na približno 19 % učnih podatkov. Pri večjih velikostih učne množice, se je algoritem velikokrat ujel v lokalni optimum, prav tako se je večala računska zahtevnost. V literaturi se za to zbirko rezultati najboljših metod gibljejo pod 40 % napake, kjer so pa izbrane arhitekture vedno ročno načrtovane (Zhang, 2017). Glede na velikost uporabljene učne množice, je naš rezultat na tej podatkovni zbirki sprejemljiv.

6 Zaključek

V članku smo predstavili algoritem diferencialne evolucije za iskanje optimalnih topologij globokih nevronskih mrež na problemu razvrščanja slik. Algoritem smo preizkusili nad dvema podatkovnima zbirkama za razvraščanje slik. Rezultati so pokazali, da je možno z uporabo evolucijskega algoritma razviti oz. poiskati globoko

mrežo, ki dosega primerljive rezultate s stanjem tehnike brez uporabe poglobljenega ekspertnega znanja.

V prihodnosti bomo algoritem preizkusili še na drugih podatkovnih zbirkah, ga skušali izboljšati v smislu računske zahtevnosti, kjer bi lahko uporabili prenosno učenje. Med drugim bomo še poskušali optimizirati parametre učnega algoritma, ki ima velik vpliv na potek učenja.

Viri in literatura

Back, T. a. (1997). Handbook of evolutionary computation. Release.

Brest, J. a. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE transactions on evolutionary computation, 646-657.

He, K. a. (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, (str. 770-778).

Hinton, A. K. (2009). Learning multiple layers of features from tiny images. Toronto: Toronto, ON, Canada. Pridobljeno iz http://www.cs.toronto.edu/~kriz/cifar.html

Huang, G. a. (2017). Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition, (str. 4700-4708).

Kaelbling, L. P. (1996). Reinforcement learning: A survey. Journal of artificial intelligence research, 237-285.

LeCun, Y. a. (2015). Deep learning. Nature, 436-444.

Mlakar, U. a. (2017). Multi-objective differential evolution for feature selection in facial expression recognition systems. Expert Systems with Applications, 129-137.

Mollahosseini, A. a. (2019). AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild. IEEE Transactions on Affective Computing, 18-31.

Sun, K. H. (2015). Deep Residual Learning for Image Recognition. arXiv.

Weinberger, G. H. (2018). Densely Connected Convolutional Networks. arXiv.

Zhang, Y. a. (2017). Very deep convolutional networks for end-to-end speech recognition. 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP), (str. 4845-4849).

Zisserman, K. S. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv, 2015.

O avtorju

Uroš Mlakar je leta 2019 dokotoriral na Univerzi v Mariboru. Trenutno je izvoljen v naziv docent za področje računalništvo na Fakulteti za elektrotehniko in računalništvo v laboratoriju za sistemsko programsko opremo. Sodeloval je že pri pripravi več kot 30 znanstvenih revijalnih in konferenčnih člankov. Njegovi raziskovalni interesi vključujejo evolucijsko računanje, podatkovno rudarjenje in obdelavo slik. Med drugim je tudi recenzent pri mnogih priznanih mednarodnih revijah.