

Nejc Podvratnik

**UČENJE REDKIH NEVRONSKIH MREŽ Z ITERATIVNIM REZANJEM PARAMETROV**

Magistrsko Delo

Maribor, avgust 2024



Nejc Podvratnik

**UČENJE REDKIH NEVRONSKIH MREŽ Z ITERATIVNIM REZANJEM PARAMETROV**

Magistrsko delo

Maribor, avgust 2024

**UČENJE REDKIH NEVRONSKIH MREŽ Z ITERATIVNIM REZANJEM PARAMETROV**

**Magistrsko delo**

|  |  |
| --- | --- |
| Študent: | Nejc Podvratnik |
| Študijski program: | Študijski program 2. stopnje  Računalništvo in informacijske tehnologije |
| Mentor: | izr. prof. dr. Damjan Strnad |
| Somentor: | Štefan Horvat, mag. inž. rač. in inf. tehnol. |
| Lektorica: | Ime in priimek, naziv |
|  |  |
|  | About CC Licenses - Creative Commons |

**ZAhvala**

Zahvaljujem se mentorju izr. prof. dr. Damjanu Strnadu

in somentorju Štefanu Horvatu, mag. inž. rač. in inf. tehnol.

za pomoč in usmerjanje pri pisanju magistrskega dela.

Posebna zahvala tudi družini, ki mi je

omogočila študij in mi stala ob strani.

Učenje redkih nevronskih mrež z iterativnim rezanjem parametrov

**Ključne besede:** strojno učenje, redka nevronska mreža, hipoteza loterijski srečk, rezanje

**UDK:** xxxxxxxx

**Povzetek**

*Moderno globoko učenje pogosto vključuje nevronske mreže, ki imajo nepotrebno veliko število parametrov oziroma povezav. Posledica tega je višja časovna in prostorska zahtevnost pri delu z mrežami. Možna rešitev problema je hipoteza loterijskih srečk, ki pravi da v množici povezav vsake naprej povezane nevronske mreže obstaja podmnožica, ki je manjša in ohranja enako in v nekaterih primerih tudi večjo uspešnost. Imenujemo jo zmagovita srečka. V vsebini tega dela bomo predstavili in dokazali hipotezo loterijske srečke, implementirali lasten algoritem za iterativno rezanje parametrov in ga testirali ter analizirali na raznih arhitekturah, podatkovnih zbirkah in hiperparametrih.*

Sparse neural network learning with iterative parameter removal

**Keywords:** machine learning, sparse neural network, lottery ticket hypothesis, pruning

**UDK:**  xxxxxxxxx

**Abstract**

*Modern deep learning often involves neural networks that have an unnecessarily large number of parameters or connections. This results in a higher time and space complexity when working with networks. A possible solution to the problem is the lottery ticket hypothesis, which states that in a set of connections of any feedforward neural network there is a subset that is smaller and maintains the same and, in some cases, even higher effectiveness. We call them winning tickets. In the content of this work, we will present and prove the lottery ticket hypothesis, implement our own algorithm for iterative pruning of parameters and test and analyze it on various architectures, datasets and hyperparameters.*

Slika, ki vsebuje besede besedilo

Opis je samodejno ustvarjen

Kazalo vsebine

[Kazalo slik X](#_Toc178871665)

[Kazalo Tabel X](#_Toc178871666)

[Uporabljeni simboli in kratice XI](#_Toc178871667)

[1 Uvod 1](#_Toc178871668)

[2 Osnovni POjmi in metodologija 3](#_Toc178871669)

[2.1 Nevron 3](#_Toc178871670)

[2.2 Nevronska Mreža 3](#_Toc178871671)

[2.3 Arhitekture Mrež 3](#_Toc178871672)

[2.4 Učenje 3](#_Toc178871673)

[2.4.1 Poglavje 2 3](#_Toc178871674)

[2.5 Hipoteza loterijskih srečk 3](#_Toc178871675)

[3 IMplementacija 4](#_Toc178871676)

[3.1 Podatkovni zbirki 4](#_Toc178871677)

[3.1.1 Klasifikacija – RealWaste 4](#_Toc178871678)

[3.1.2 Regresija – X-Ray Age and Gender Dataset 6](#_Toc178871679)

[3.2 Algoritem iterativnega rezanja parametrov s hipotezo loterijske srečke 7](#_Toc178871680)

[3.2.1 Inicializacija maske 9](#_Toc178871681)

[3.2.2 Kopiranje začetnega stanja modela 10](#_Toc178871682)

[3.2.3 Zamrznitev rezanih uteži 10](#_Toc178871683)

[3.2.4 Rezanje uteži 11](#_Toc178871684)

[3.2.5 Povrnitev začetnega stanja modela 12](#_Toc178871685)

[3.3 Testiranje 13](#_Toc178871686)

[3.3.1 Poglavje 1 13](#_Toc178871687)

[3.3.2 Poglavje 2 13](#_Toc178871688)

[4 REZULTATI in analiza 13](#_Toc178871689)

[5 SKLEP 13](#_Toc178871690)

[Viri in literatura 15](#_Toc178871691)

Kazalo slik

[Slika 3.1: Primer slike iz RealWaste (levo) in TrashNet (desno) 2](#_Toc174793729)

[Slika 3.2: Primer augmentacije slike RealWaste 3](#_Toc174793730)

[Slika 3.3: Primerki iz X-Ray Age and Gender Dataset 4](#_Toc174793731)

Kazalo Tabel

[Tabela 3.1: Kategorije odpadkov in njihovo število za RealWaste 2](#_Toc174713011)

Uporabljeni simboli in kratice

SPR – Društvo za radiološko in diagnostično slikanje iz São Paula (angl. The Radiology and Diagnostic Imaging Society of São Paulo)

# Uvod

V današnjem svetu se uporaba umetne inteligence iz leta v leto raste. Uporablja se pri raznih aspektih industrije, gospodarstva in računalniških storitev. Ampak tako kot se veča uporaba, se veča tudi kompleksnost opravil, ki jo nevronske mreže morajo izvesti. Zahtevnejša opravila potrebujejo večjo oz. globljo arhitekturo, kar zahteva več računalniških virov in prostora [6]. To je še posebej vidno pri dveh primerih uporabe. Prve so v malih napravah, ki se pogosto uporabljajo pri internet stvari (angl. Internet of things), ki so premajhne, da bi lahko vsebovale velike oziroma globoke arhitekture. Druge so pa realno časovne, kot je na primer izboljšanje kvalitete predvajanja v živo s pomočjo umetne inteligence. Če predpostavimo, da se ukvarjamo s hitrostjo 60 sličic na sekundo ima računalnik okoli 16.67 milisekund časa da pošlje sliko skozi nevronsko mrežo in opravi vse ostalo, kar je zahtevano od njega. Ne pomaga to, da v praksi raje uporabimo kompleksnejši model in z dodatnimi tehnikami poskušamo zaznati in omejiti prekomerno prileganje [7]. Res je da s temi tehnikami ne bo prišlo do prekomernega prileganja ampak prostor in zahtevani računalniški viri bodo večji kot bi morali. Dobra rešitev je, da nekatere parametre oziroma povezave preprosto izrežemo (angl. prune) iz modela. To bo zmanjšalo prostor in računsko zahtevnost modela. Ampak katere povezave naj odrežemo, da ne bo uspešnost mreže padla? Obstajajo sploh takšne povezave? Odgovor na to vprašanje ima hipoteza loterijske srečke.

Prvi cilj dela je, da vam predstavimo in razložimo hipotezo loterijske srečke in kaj je zmagovita srečka. Torej kje in kako bi jo lahko uporabili v kombinaciji z iterativnim rezanjem parametrov. Na trivialnem primeru bomo dokazali da hipoteza velja. Drugi cilj pa je da implementiramo lastni program, ki bo s pomočjo hipoteze loterijske srečke in iterativnega rezanja parametrov, učil modele, ki bodo imeli bistveno manj aktivnih parametrov kot originalni in imeli primerljivo ali v nekaterih primerih tudi večjo uspešnost kot originalni. [TUKAJ ŠE DODAJ ŽE OBSTOJEČE ČLANKE]

V prvem poglavju se bomo dotaknili teorije. Prvo bomo na kratko razložili osnove nevrona, nevronske mreže in strojnega učenja ter pojme ki bodo pomembni za razumevanje tega dela. Dotaknili se bomo modelov, ki bodo uporabljeni pri testiranju. Nato pa se bomo podrobneje poglobili v teorijo loterijske srečke. V naslednjem poglavju bo opisana implementacija. Tukaj bomo omenili v katerem okolju smo ustvarili izdelek, katere podatkovne zbirke smo uporabili in potek programa. Nato pa bomo končan izdelek testirali in analizirali rezultate v naslednjem poglavju. V sklepu pa povzamemo delo ter omenimo katere aspekte naloge bi lahko spremenili oz. nadgradili.

# Osnovni POjmi in metodologija

Človeštvo je pogosto vzelo inspiracijo iz narave in ena izmed teh je nevronska mreža….

## Nevron

Na kratko razloži kaj je nevron, bolj specifično kaj so uteži in odmiki.

## Nevronska Mreža

Da je sestavljena iz plasti nevronov, vhodna izhodna in skrite plasti. Razloži tudi kaj je feed forward nevronska mreža. V bistvu samo razloži arhitekturo nevronske mreže

## Arhitekture Mrež

V podpoglavjih na kratko razloži arhitekture (resne, alexnet in vgg), ki jih boš uporabil pri testiranju

## Učenje

Tukaj razloži osnove učenja, torej vse parametre, kaj je under in overfittanje, kaj je early stopping …

### Poglavje 2

Tukaj na kratko razloži kaj je regresija in klasifikacija in katere izgubne funkcije se za njo uporabljajo

## Hipoteza loterijskih srečk

Zraven opisa hipoteze in vsega podobnega boš tukaj izvajal tudi nekaj učenja. V bistvu boš dokazal, da hipoteza loterijske srečke velja tako da boš izvajal naključno rezanje in primerjal rezultate. Lahko tudi pogledaš kako se bodo vrednosti uteži skozi čas spremenile.

# IMplementacija

Aplikacijo za iterativno rezanje parametrov smo napisali v programskem jeziku Python, zato ker je daleč največ knjižnic in podpore za strojno učenje. Dve osrednji knjižnici za upravljanje z nevronskimi mrežami sta Tensorflow in Pytorch. Izbrali smo Pytorch, ker je popularnejša izbira med uporabniki kot tudi raziskovalci. Leta 2021 je Pytorch bil trikrat bolj pogosto uporabljen v člankih kot pa Tensorflow [5].

Modeli v knjižnici Pytorch vsebujejo ukaz *named\_parameters*, ki vrne seznam vseh učnih parametrov. Če iz tega seznama filtriramo samo uteži, dobimo seznam tenzorjev za vsako plast. To nam omogoča preprost dostop in manipulacijo posameznih uteži, kar je pomembno za našo nalogo.

## Podatkovni zbirki

V sklopu tega dela bomo izvajali strojno učenje na dveh podatkovnih zbirkah (angl. *datasets*), vsak za en namen uporabe. Za klasifikacijo bomo uporabili zbirko RealWaste, za regresijo pa SPR (angl. The Radiology and Diagnostic Imaging Society of São Paulo ) X-Ray Age and Gender Dataset. Obe zbirki smo razdelili na učno, validacijsko in testno množico z delitvijo 70-15-15 in jih pred učenjem normalizirali na vrednosti med 0 in 1.

### Klasifikacija – RealWaste

Realwaste je bila prvič predstavljena v članku z istim imenom [1]. Namenjena je za učenje nevronskih mrež za ločevanje odpadkov na smetišču. Od podobnih podatkovnih zbirk, kot je TrashNet [2], se razlikuje po slikanih smeteh. Odpadki v zbirki RealWaste so slikani na odpadu in zato vsebujejo različne deformacije in umazanije, ki jih embalaža slikana doma nima, kar povečuje njeno efektivnost (slika 3.1).



Slika .: Primer slike iz RealWaste (levo) in TrashNet (desno)

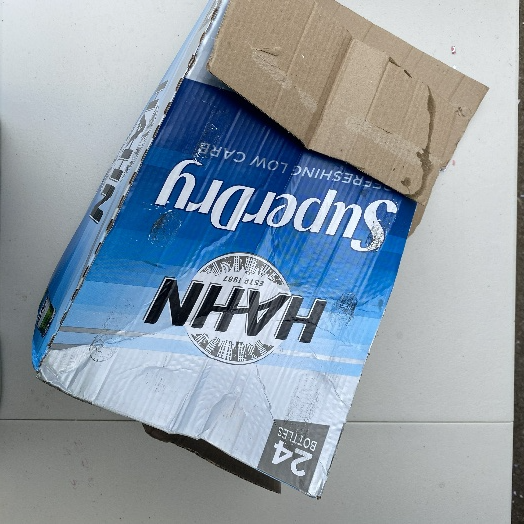
Vsebuje 4752 barvnih slik. Vsaka slika vsebuje natančno en odpadek, ki je postavljen na sredino. Odpadki so razdeljeni v 9 skupin, glede na vrsto materiala (tabela 3.1).

|  |  |
| --- | --- |
| Kategorije | Število slik |
| Karton | 461 |
| Organska hrana | 411 |
| Steklo | 420 |
| Kovina | 790 |
| Ostali odpadki | 495 |
| Papir | 500 |
| Plastika | 921 |
| Tekstil | 318 |
| Rastlinstvo | 436 |

Tabela .: Kategorije odpadkov in njihovo število za RealWaste

Tako kot pri izvirnem članku, bomo pred učenjem izvedli dve strategiji: velika velikost slik in povečanje učne množice (angl. training set augmentation). Slike bodo velikosti 524x524, kar bo med učenjem pomagalo pri ločevanju raznih podrobnostih smeteh in s tem povišalo uspešnost nevronske mreže. Na primer večja slika bo pomagala pri ločevanju med plastenkami in steklenicami [3]. Učno množico smo povečali na dva preprosta načina: horizontalno zrcaljenje slike in rotacija za 90 stopinj v smeri urinega kazalca (slika 3.2). S tem smo potrojili velikost učne množice. Validacijski in testni množici sta pa ostali nespremenjeni.

Slika .: Primer augmentacije slike RealWaste



### Regresija – X-Ray Age and Gender Dataset

Podatkovna zbirka SPR X-Ray Age and Gender se nahaja na spletni strani Kaggle [4]. Bila je del tekmovanja, kjer so uporabniki tekmovali katera nevronska mreža je čim bolj natančno ocenila starost in spol osebe glede na njeno rentgensko sliko prsnega koša. Uporabili bomo samo regresijski del podatkovne zbirke, torej samo starost. Tako kot večina tekmovalcev smo slike zmanjšali iz 1024x1024 na 256x256 za veliko hitrejše učenje. Zbirka vsebuje 10702 sivinskih slik s starostjo kot njena ciljna vrednost (slika 3.3). Pretvoriti smo jih morali v barvne slike, da bi se prilegale nevronskim mrežam, ki zahtevajo tri kanale za vhod.

Slika, ki vsebuje besede rentgenski film, medicinsko slikanje, radiologija, radiografija

Opis je samodejno ustvarjen

Slika .: Primerki iz X-Ray Age and Gender Dataset

## Algoritem iterativnega rezanja parametrov s hipotezo loterijske srečke

Algoritem prejema več (hiper)parametrov:

* NUM\_OUTPUTS – število izhodov, ki jih bo model imel. Regresija ima samo en izhod, klasifikacija pa toliko kolikor razredov ima podatkovna zbirka.
* BATCH\_SIZE – velikost minipaketa
* NUM\_PRUNE\_ITERATIONS – koliko iteracij rezanja parametrov se bo zgodilo
* PRUNE\_PER – kolikšen delež aktivnih parametrov želimo rezati (npr. z 0.2 bomo v vsaki iteraciji izrezali 20 odstotkov aktivnih parametrov)
* IMAGE\_SIZE – velikost slik
* LR – hitrost učenja
* patience – kako hitro se naj učenje modela zaključi, če se učenje ni več koristno

Pri razlagi implementacije ne bomo omenili kako smo pripravili učno množico, inicializirali uteži, implementirali učenje in ostale pogoste aspekte strojnega učenja. Razložili bomo samo tiste dele programa, ki se nanašajo na temo dela.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, diagram

Opis je samodejno ustvarjen

Slika .: Diagram poteka za iterativno rezanje parametrov

Grob potek programa je vizualno predstavljen z diagramom potema na sliki 3.4. Nekateri koraki bodo podrobneje opisani v naslednjih poglavjih. Program ima dve zanki. V manjši poteka normalno učenje modela z enim dodatkom. Preden posodobimo nove vrednosti uteži nastavimo gradient vseh rezanih uteži na 0. To povzroči da se rezane uteži ne spremenijo in tako ostanejo »zamrznjene«. Učenje nima nastavljenega fiksnega števila epoh. Konča se kadar učenje modela ni več koristno, kar je vidno s spremembo napake nad validacijsko množico. Implementirali smo razred, ki izvaja zelo podobno nalogo predčasni prekinitvi učenja (angl. early stoppage). Med učenjem si zapomni najnižjo doseženo validacijsko napako in na disk shrani začasno datoteko stanja modela, ki je dosegel to napako. Če v določenih korakih (tj. parameter *patience*) model ni dosegel nižje validacijske napake, kot trenutne najnižje, se trenutno stanje modela nadomesti s shranjenim stanjem in prenehamo učenje.

Večja zanka vsebuje manjšo. Po učenjem testiramo naučen model nad testno množico. Model shranimo in v imenu napišemo kolikšen odstotek uteži je bilo aktivnih ter njegovo uspešnost nad testno množico. Nato izberemo katere uteži bomo rezali iz modela. Na koncu povrnemo vrednosti uteži modela na njegovo začetno vrednost, izrezane uteži pa nastavimo na 0. S tem se konča ena iteracija rezanja parametrov. Pri vsaki iteraciji se število aktivnih uteži modela zmanjša. V primeru da v eni iteraciji izrežem 20% uteži bomo v prvi iteraciji začeli s 100% aktivnih uteži nato s 80%, 64%, 51.2%, 40.96% itd.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, številka

Opis je samodejno ustvarjen

### Inicializacija maske

Pred učenjem definiramo spremenljivko, ki ji pravimo maska. Vsebovala bo informacije katera utež je aktivna in katera je izrezana. Aktivne uteži so pred učenjem naključno generirane in se posodabljajo med učenjem, izrezane uteži pa so pred učenjem nastavljene na 0 in se ne posodabljajo med učenjem. Maska je seznam tenzorjev, ki je enake oblike kot seznam tenzorjev uteži. Vsebuje samo vrednosti 0 in 1. Če je vrednost elementa v maski enaka 1, potem je utež na enaki lokaciji v pripadajoči plasti aktivna. Drugače je izrezana. Pri inicializaciji maske se sprehajamo po plasteh nevronske mreže, od vhodne do izhodne plasti. Za vsako plast ustvarimo nov element v seznamu in mu določimo enako obliko kot jo imajo uteži. Na začetku izvajanja so vse vrednosti nastavljene na 1.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, vrstica

Opis je samodejno ustvarjen

### Kopiranje začetnega stanja modela

Če želimo da bo iterativno rezanje parametrov s hipotezo loterijske srečke uspešno, moremo po vsakem rezanju vrniti vrednosti aktivnih uteži na njihovo začetno vrednost. Zato pred učenjem kopiramo začetno stanje modela, tj. vse uteži in odmike.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, vrstica

Opis je samodejno ustvarjen

### Zamrznitev rezanih uteži

Pytorch ima funkcionalnost, da prepreči popravljanje uteži celotne plasti ampak ne specifične uteži. Zato smo morali uteži zamrzniti ročno. Na koncu iteracije učenja se izračunajo gradienti za vsako utež. Te gradiente pri optimizacijskem koraku uporabimo pri izračunu popravka uteži (). Če gradiente zamrznjenih uteži nastavimo na 0, bo tudi popravek posledično 0 in s tem bo utež nespremenjena. Na srečo imajo gradienti enako obliko kot uteži. Zato lahko izvedemo Hadamardov produkt (tj. množenje isto ležečih elementov) nad gradienti in masko in s tem dobimo nov tenzor in ga zamenja s trenutnim tenzorjem gradientom.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, številka

Opis je samodejno ustvarjen

### Rezanje uteži

Pri tem koraku izrežemo določen odstotek najmanj pomembnih uteži iz modela. Pri naši implementaciji izračunamo prag, ki loči aktivne od izrezanih uteži. Vse vrednosti uteži, ki so večje od praga ostanejo aktivne, manjše so pa izrezane. Ideja je, da vse vrednosti uteži razvrstimo od najmanjšega do največjega po njegovi absolutni vrednosti. Prag se nahaja na istem odstotku poti do konca polja, kolikor odstotkov uteži želimo izrezat. Torej če želimo izrezati 20% aktivnih uteži, se prag nahaja na indeksu.

Na začetku vse tenzorje uteži iz vseh plasti preoblikujemo in združimo v veliko enodimenzionalno polje, kjer ignoriramo že izrezane uteži. Nato jih spremenimo v njihove absolutne vrednosti in jih razvrstimo po indeksih (angl. argsort) od najmanjšega do največjega. Indeks, ki se nahaja na določenih odstotkih poti je vsebuje prag. Na koncu posodobimo masko tako, da pogledamo absolutno vrednost isto ležeče uteži, ki ga element maske predstavlja. Element maske nastavimo na 0, če je absolutno vrednost uteži manjša od praga. V primeru če je večja, elementa maske ne spreminjamo.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, dokument

Opis je samodejno ustvarjen

OMENI POSEBNA PRAVILA ZA VSE ARHITEKTURE

### Povrnitev začetnega stanja modela

V zadnjem koraku v zanki pripravimo model za učenje v naslednji iteraciji. Izvedemo Hadamardov produkt nad začetnim stanjem modela in masko, ki je v prejšnjem koraku bila posodobljena. Izhod je novo stanje modela, kjer so izrezane uteži nastavljene na 0, aktivne pa na njihovo začetno stanje.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, vrstica

Opis je samodejno ustvarjen

## Testiranje

Tukaj napiši glavni namen tega testiranja. To je ugotoviti koliko odstotkov uteži lahko režemo v eni iteraciji.

### Poglavje 1

Tukaj podrobneje opiši testiranje. Torej katere parametre in modele si uporabil in testiral

### Poglavje 2

Arhitekture resnet18, alexnet in vgg11 imajo več milijon učnih parametrov, ki niso enakomerno razdeljeni med vsemi plastmi. Razlika med številom učnih parametrov je lahko ogromna. Nekatere plasti imajo par sto uteži, nekatere pa par milijon. Lahko se zgodi da v manjših plasteh izrežemo preveč uteži, kar bi ustvarilo ozko grlo za celotno arhitekturo in s tem drastično znižalo uspešnost nevronske mreže. Pri uporabljenih arhitekturah se ta problem lahko pojavi kljub globalnemu načinu rezanja, zlasti pri višjem odstotku rezanja in nižjem odstotku aktivnih uteži. Zato smo algoritem za izrezovanje uteži spremenili tako, da izpusti manjše uteži.

Pri vgg11, ki ima približno 128 milijonov uteži, smo izpustili prvo konvolucijsko plast s 1728 uteži in edino polno povezano plast modela s 4096 uteži za regresijo in 40960 uteži za klasifikacijo.

Pri alexnet, ki ima približno 57 milijonov uteži, smo izpustili prvo konvolucijsko plast s 23232 uteži in zadnjo polno povezano plast s 4096 uteži za regresijo in 40960 uteži za klasifikacijo.

Pri resnet18, ki ima približno 11 milijonov uteži, smo izpustili vse plasti paketne normalizacije s skupno 5056 uteži in edino polno povezano plast modela s 512 uteži za regresijo in 5120 uteži za klasifikacijo.

# REZULTATI in analiza

# SKLEP

**TODO:**

* V uvodu omeni druge članke ki imajo podobno temo
* V implementaciji napiši še posebna pravila za posamezni model pri izreži uteži
* V implementaciji še napiši kako si se ukvarjal z odmiki

Viri in literatura

[1] Single, S., Iranmanesh, S., Raad, R. *RealWaste: A Novel Real-Life Data Set for Landfill Waste Classification Using Deep Learning*. 2023. [<https://doi.org/10.3390/info14120633>]

[2] Yang, M., Thung, G. *TrashNet*, Github. 2016. Dostopno na [[**https://github.com/garythung/trashnet**](https://github.com/garythung/trashnet)**]** [16.8.2024].

[3] Sabottke, C.F., Spieler, B.M. *The Effect of Image Resolution on Deep Learning in Radiography*. Radiol. Artif. Intell. 2020. [<https://pubs.rsna.org/doi/full/10.1148/ryai.2019190015>]

[4] SPR X-Ray Age and Gender Dataset [<https://www.kaggle.com/datasets/felipekitamura/spr-x-ray-age-and-gender-dataset>][17.8.2024]

[5] AssemblyAI – Pytorch vs Tensorflow [<https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/>] [17.8.2024]

[6] Thompson, N. C., Greenewald, K., Lee, K., Manso, G. F. *The Computational Limits of Deep Learning.* 2022.[<https://arxiv.org/abs/2007.05558>]

[7] Strnad, D. *Napredne arhitekture in učenje globokih nevronskih mrež: Prekomerno prileganje*. 2024.

Slika, ki vsebuje besede besedilo, posnetek zaslona, kolaž, mozaik

Opis je samodejno ustvarjen