Klasifikacija prometnih znakova konvolucijskom neuronskom mrežom učenom na reduciranom skupu podataka

Lipanj 2021.

Domagoj Bošnjak
Matematički odsjek
Prirodoslovno-matematički fakultet
Zagreb, Hrvatska
dombosn@student.math.hr

Iskra Gašparić
Matematički odsjek
Prirodoslovno-matematički fakultet
Zagreb, Hrvatska
iskgasp@student.math.hr

***Sažetak*—Problem pravilne klasifikacije prometnih znakova izrazito je važan za moderne sustave potpuno ili djelomično autonomnih vozila. Standardni pristupi takvoj klasifikaciji uključuju korištenje *konvolucijskih neuronskih mreža*. Nadalje, uobičajeno je na različite načine povećavati skup podataka. S druge strane, postavlja se prirodno pitanje -može li se katkad *smanjenjem skupa podataka* postići aproksimativno *ista točnost* uz relativno značajno *smanjenje vremena treniranja*. Upravo će to biti istraženo kroz ovaj rad: prezentiran je dataset i njegovo preprocesiranje, postupak redukcije podataka korištenjem klasteriranja i pripadnih značajki te konačno konkretan primjer CNN-a. Na samom kraju demonstriraju se rezultati s naglaskom na trajanje i točnost klasifikacije.**

I. Uvod

Popularizacijom koncepta autonomnih vozila i razvojem ADAS-a (advanced driver assistance systems) u svrhu što veće sigurnosti u prometu, javlja se potreba za kvalitetnom detekcijom prometnih znakova. Glavni fokus ovog rada bit će njihova klasifikacija. Uz eventualni, generalno preporučeni, *data augmentation*, dolazimo do poprilično velikog broja primjera za treniranje. To je samo po sebi korisno, ali uvjetuje sporu manipulaciju podatcima. Ukoliko, primjerice, odlučimo promijeniti neki od parametara u modelu, moramo ponovno pokrenuti proces učenja koji je to sporiji i memorijski zahtjevniji što je skup veći. Dodatno, razumna je pretpostavka da za željenu preciznost nisu nužni svi dani primjeri. Stoga je osim prirodnog zahtjeva točnosti klasifikacije, još jedan cilj ovog rada svojevrsni *data reduction*, odnosno pokušaj odbacivanja dijela primjera iz skupa za treniranje uz minimalan ili nikakav gubitak točnosti. Time bi se zadržali informativni primjeri iz skupa za treniranje, a maknuli oni redundantni, koji potencijalno mogu dovesti do overfittinga i povećanja vremena treniranja. Primjer sličnog postupka i detaljnija motivacija može se pronaći u [1]. Skup podataka koji se koristi jest GTSRB skup, koji sadrži preko 50000 slika prometnih znakova, a plan je ne koristiti taj cijeli (relativno veliki) skup za treniranje. Glavna metoda rješavanja problema jest konvolucijska neuronska mreža, potaknuta dodatno činjenicom da se u top 10 metoda na natjecanju iz kojeg dataset originalno potječe našlo nekoliko konvolucijskih neuronskih mreža. Povrh toga, konvolucijske neuronske mreže pokazale su se izrazito efikasne u području precizne klasifikacije slika.

II. Hipoteze

Učenje mreže na reduciranom skupu podataka očitovat će se u smislu efikasnosti, a potencijalno i preciznosti. Parametre u treniranju mreže trebat će prilagoditi problemu što rezultira višestrukim provođenjem procesa s različitim parametrima. Tako ćemo već u izradi drugog dijela projekta (učenju modela) opravdati proces redukcije podataka smanjenjem potrebnog vremena. Nasumično micanje podataka iz skupa za treniranje postiglo bi isti cilj u smislu vremena, ali nema apsolutno nikakve garancije da se točnost mreže neće smanjiti, čime bi podešavanje parametara izgubilo smisao. Stoga je finalni cilj postići dovoljno točnu neuronsku mrežu za prepoznavanje prometnih znakova, uz uvjet da mreža ne koristi cijeli originalni skup za treniranje.

III. Skup podataka *GTSRB*

Dataset *German Traffic Sign Recognition Benchmark* (nadalje GTSRB) prvi puta je korišten na pripadnom natjecanju u klasifikaciji održanom $2011.$ godine. Skup sadrži 51839 slika prometnih znakova različitih dimenzija (najmanje dimenzije su 25×25). Podijeljen na skup za treniranje i skup za testiranje te je sve skupa dostupno je javno na linku [2]. Na istom linku dostupni su i najbolji radovi na natjecanju.

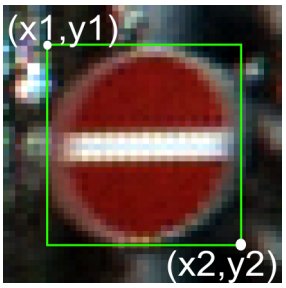
Znakovi su podijeljeni u 43 klase, predstavljene kao brojevi od 0 do 42. Svakoj slici pridružene su vrijednosti širine, visine, koordinate gornjeg lijevog $(x1, y1)$ i donjeg desnog kuta $(x2, y2)$ znaka unutar slike kao na Slici 2 te oznaka klase (*ClassId*).

IV. Metode i implementacija

A. Preprocesiranje slika

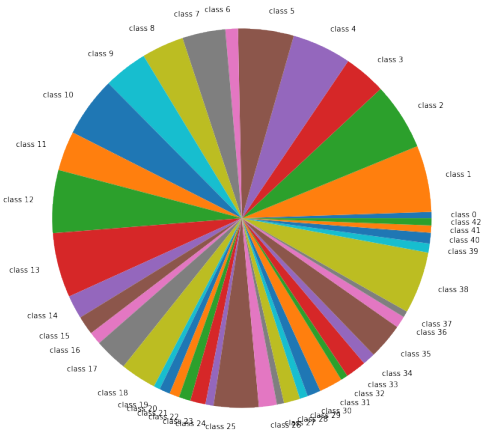
Budući da su slike različitog formata, najprije su sve skalirane na dimenzije 30×30. Za algoritam reduciranja

Slika 1. Primjeri slika iz GTSRB dataseta



Slika 2. Primjer oznake slike iz GTSRB dataseta

podataka nužno je da slike budu crno-bijele (2D objekti) pa pohranjujemo crno-bijele slike u posebno polje. Primijetimo da raspored slika po klasama nije ujednačen:



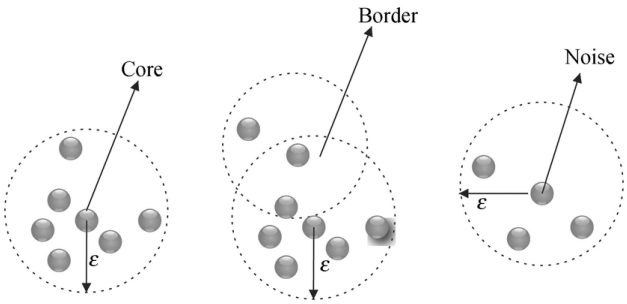
Slika 3. Odnos veličina različitih klasa

Zbog toga bi bilo povoljno napraviti augmentaciju podataka na klasama za koje je dostupan relativno malen broj slika. Isprobane su obje varijante - sa i bez augmentacije te su rezultati navedeni u nastavku rada.

B. Redukcija podataka

Intuicija iza reduciranja podataka je sljedeća: ako s K elemenata možemo (gotovo) savršeno klasificirati ostalih $N - K$ elemenata u danom skupu podataka, prirodno je pitanje je li nužno imati svih N elemenata. Konkretan postupak svodi se na uklanjanje "sličnih" primjeraka što kao prirodan odabir metode nameće klasteriranje. Iz svakog klastera uzima se jedan element, a u skupu zadržavamo i sve slike koje nisu pripale nijednom klasteru (*noise points*, odnosno svojevrsni *outlieri*), budući da takve slike evidentno nude u nekoj mjeri jedinstvene informacije. Dodatno, klasteriranje i selekcija provode se unutar svake od 43 klase neovisno o ostalim klasama.

Konkretan algoritam za klasteriranje korišten u implementaciji je *OPTICS* (Ordering Points To Identify the Clustering Structure) algoritam koji je dostupan unutar biblioteke *scikit-learn*, a originalno dan u [8]. Sam algoritam nastao je kao svojevrsno produženje DBSCAN algoritma. Adresira jednu od njegovih najvećih slabosti - detektiranje klastera u datasetu varijabilne gustoće. *OPTICS* algoritam pripada skupini *density-based* algoritama za klasteriranje, temeljenih na pronalasku mjesta visoke gustoće unutar dataseta. Prolaskom po elementima detektiraju se elementi koji u okolini radijusa ε sadržavaju $min\_samples$ objekata, to jest, kardinalitet okoline mora biti veći ili jednak zadanoj vrijednosti $min\_samples$. Takve elemente zovemo *core elementima*. Ukoliko neki element sam po sebi nije core element, ali u svojoj okolini sadrži barem jedan core element, zovemo ga *border* element. Konačno, elemente koji nisu niti core niti border elementi, odnosno ne formiraju klaster niti imaju susjedni element koji formira klaster, algoritam će označiti kao *noise point*. Jednostavan primjer klasifikacije elemenata dan je na slici ispod.



Slika 4. Primjeri core, border odnosno noise elemenata

Upravo su *border* elementi izvor nejedinstvenosti klasteriranja OPTICS algoritmom. Ukoliko je element klasificiran kao takav, a u svom ε-susjedstvu ima više *core* elemenata, bit će pridružen onom klasteru koji ga je "prvi pronašao".

Dodatno, parametar ε nije toliko utjecajan kao kod DBSCAN algoritma pa ga niti nije nužno specificirati, no parametar $min_samples$ je izrazito bitan. Stoga je prirodno pitanje kako robusno odrediti min_samples parametar, odnosno kako procijeniti je li klasteriranje s odabranim parametrom adekvatno.

Glavna evaluacija kvalitete reduciranja podataka bit će točnost mreže te njeno vrijeme treniranja u odnosu na iste vrijednosti prije redukcije. No, postoje i mnoge mjere za evaluaciju kvalitete klasteriranja, koje ne ovise o podatcima. Koncizan pregled tzv. *internih mjera* kvalitete klasteriranja dan je u [6]. U ovom radu primarno će se koristiti *S_dbw* mjera, detaljnije opisana u [7], dostupna kao vlastiti Python module.

Iako ponešto vremenski izazovno rješenje, moguće je provesti cijeli postupak te provjeriti koji izbor parametra

min_samples daje najbolju vrijednost mjere kvalitete klasteriranja *S_dbw*. Zanimljiva primjedba je da su za izbor parametra min_samples u skupu $\{2, \ldots, 10\}$ sva klasteriranja dala poprilično dobar S_dbw rezultat, ali ipak se najbolji rezultat uglavnom dobivao za min_samples $= 2$.

U postupku rekdukcije podataka slike se ne uzimaju kao takve, nego ih reprezentiramo pomoću *Haar-like* značajki.

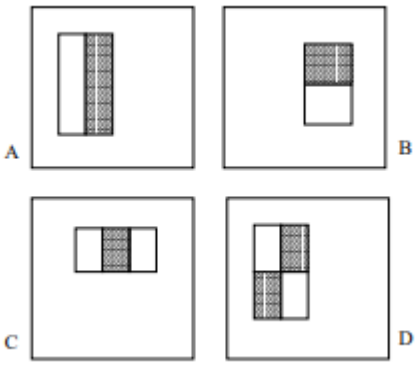
C. Haar-like značajke

Haar-like značajke, originalno dane u [4], služe za detektiranje raznih uzoraka na slikama, primjerice vertikalnih i horizontalnih rubova, dijagonala, oblika poput $n \times n$ šahovske ploče i slično. Jedna od njihovih glavnih prednosti jest ta da se nakon jednog računanja tzv. *integralne slike* svaka značajka računa u konstantnom vremenu.

1) Integralna slika: Označimo sa $ii(x, y)$ integralnu sliku, tj. njenu vrijednost na poziciji (x, y). Vrijednosti piksela u originalnoj slici označavamo sa $i(x, y)$. Integralna slika na poziciji (x, y) sadrži sumu vrijednosti piksela od gornjeg lijevog kuta originalne slike do pozicije (x, y) uključivo:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

Jasno je da se integralna slika može izračunati jednim prolazom po slici, budući da se već izračunata vrijednost $ii(x, y)$ može iskoristiti za računanje $ii(x', y'), x' \geq x, y' \geq y$.



Slika 5. Primjeri računanja Haar-like značajki

Dakle, nakon jednog računanja integralne slike svaka se značajka računa u konstantnom $(O(1))$ vremenu. Dodatno, implementacija račnunanja Haar-like značajki dostupna je u *scikit-learn* biblioteci. Konačno, prije prelaska na diskusiju o konvolucijskoj neuronskoj mreži, dan je sažetak algoritma za redukciju podataka. Slike koje su dodijeljene klasteru dobivaju oznaku prirodnog broja tog klastera, dok je oznaka noise point slika -1 pa se zbog toga slike označene s -1 zadržavaju.

Prvi dio algoritma u kojemu se bira optimalan min_samples parametar je zapravo opcionalan. Nudi svojevrsnu robusnost na manje kvalitetna klasteriranja, u zamjenu za vremenski zahtijevniju provedbu redukcije podataka.

Algoritam 1: Redukcija podataka

```
Input: preprocesirane slike po klasama
  S_i, i = 1,...,43, lista P min_samples parametara,
  matrica značajki F
Keep_images = [ ]
for i ∈ {1,...,43} do
    for p ∈ P do
        k = OPTICS(F, min_samples = p)
        m = S_dbw(k)
        if m < optimal_m then
            optimal_m = m
            optimal_k = k
    labels = set(optimal_k.labels)
    for image ∈ S_i do
        label = labels(image)
        if label == -1 then
            Keep_images.append(image)
        if label != -1 and label ∈ labels then
            Keep_images.append(image)
            labels.remove(label)
```

2) Odabir značajki: Način na koji funkcionira *scikit-learn* funkcija za računanje Haar-like značajki jest da kao argument uzme gornji lijevi kut prozora na slici te njegova širina i visina, nakon čega se izračunaju sve moguće Haar-like značajke unutar tog prozora. Jasno, taj se postupak izvodi jednako za svaku sliku iz skupa. Značajke izračunate u eksperimentima većinski su fokusirane na sredinu slike, budući da su informacije na osnovu kojih želimo provesti diferencijaciju slika uglavnom na tom području slike. Korištena su sva tri tipa značajki demonstrirana na slici 5. Dodatno, značajke na slici 5 u gornjem redu zovu se značajke *tipa 2*, slika C prikazuje značajke *tipa 3*, a značajke u obliku šahovske ploče zovemo značajke *tipa 4*.

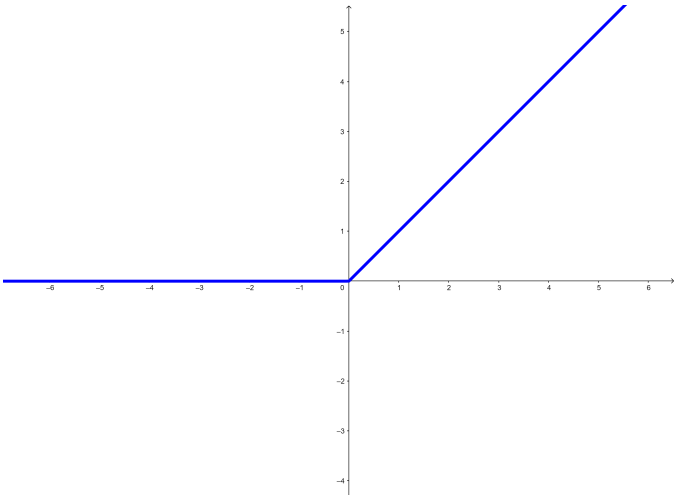
Nadalje, u nekim primjerima proveden je i *feature selection*. Konkretnije, koristi se *univarijatna metoda*, koja iz matrice značajki statističkim univarijatnim testovima bira određenu količinu *najbitnijih* značajki. U tom pogledu *scikit-learn* nudi `SelectPercentile(f, p)` funkciju, koja na osnovu neke funkcije f odabere najboljih p posto značajki. Za problem klasifikacije preporučeno je (za funkciju f) odabrati *scikit-learn* funkciju `f_classif`, koja računa ANOVA F-vrijednost u svrhu biranja značajki.

D. Konvolucijska neuronska mreža

Problem klasifikacije slika rješavamo konvolucijskom neuronskom mrežom. Generalni koncenzus jest da je kod multiclass klasifikacije korištenjem CNN-a dobar izbor loss funkcije *cross-entropy loss*. Glavni način evaluacije finalne točnosti jest koliko znakova iz skupa za testiranje je točno klasificirano (Train & Test metoda evaluacije modela). Implementacija neuronske mreže provedena je pomoću *Pytorch*-a.
Treniranje konvolucijske neuronske mreže i optimalni izbor parametara očekivano su zahtijevali najviše vremena. Uz

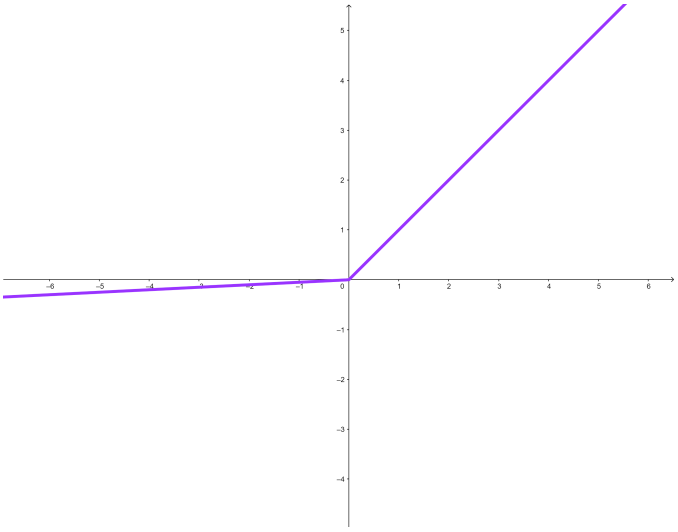
različit izbor broja skrivenih slojeva, broja epoha, aktivacijske funkcije itd. najčešće ostvarujemo točnost na skupu za testiranje od barem 94%. Kao najbolji izbor aktivacijske funkcije pokazala se $ReLU$ funkcija,

$$f(x) = max(0, x)$$



Slika 6. ReLU aktivacijska funkcija

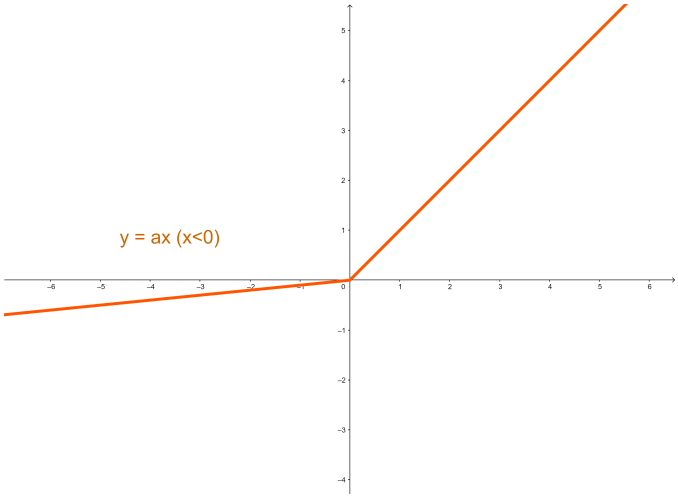
Alternativno, testirane su i *LeakyReLU* i *PReLU* aktivacijske funkcije. *LeakyReLU* vrijednosti $x < 0$ šalje u $0.01x$, radije nego 0, kao što to čini ReLU.



Slika 7. Leaky ReLU aktivacijska funkcija

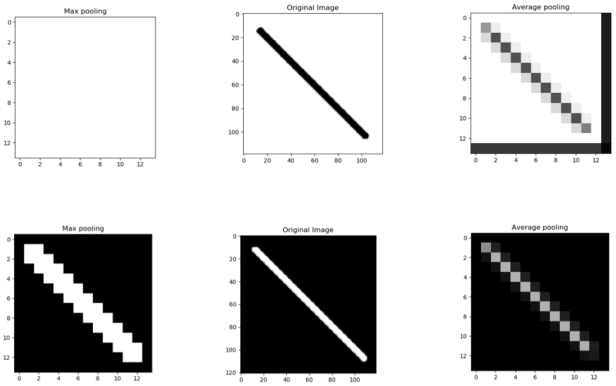
S druge strane, *PReLU*, odnosno *Parametrized ReLU* funkcija temelji se na tome da se vrijednosti $x < 0$ šalju u ax gdje se parametar a određuje pomoću same mreže.

Ipak, njihovo korištenje nije dovelo do bolje točnosti klasifikacije u odnosu na uobičajenu ReLU funkciju pa je ona i korištena.



Slika 8. PReLU aktivacijska funkcija

Pooling sloj služi za reduciranje dimenzija slike kako bi se olakšale daljnje računske operacije. Odabir funkcije za *pooling* u pravilu podosta ovisi o skupu podataka. Dvije najčešće korištene funkcije su *MaxPool* i *AvgPool* funkcije. *MaxPool* za kvadrate odabrane veličine (često 2×2) kao izlaznu vrijednost ima maksimum vrijednosti u svim poljima. Na taj način do izražaja dolaze samo najznačajniji dijelovi slike. Međutim, na taj način može se izgubiti dio značajnih informacija. Na Slici 9 preuzetoj iz [9] na dva jednostavna primjera opisana je ta pojava. Ukoliko je crna linija na posve bijeloj podlozi dovoljno tanka da je maksimum vrijednosti u svakom 2×2 bloku jednak 256 (što odgovara bijeloj boji), ta linija će se izgubiti.



Slika 9. Odabir *pooling* funkcije

S druge strane, *AvgPool* računa prosječnu vrijednost odabranih polja. Svojstvo *AvgPool* jest naglašavanje svih elemenata slike čime će se izbjeći upravo opisani efekt. Međutim, njena mana je nemogućnost naglašavanja najvažnijih značajki jer se u obzir uzimaju sve značajke u jednakoj mjeri. Taj efekt ponovno se može vidjeti na Slici 9. Jedan validan zaključak bio bi da je *MaxPool* korisnija na slikama s tamnom pozadinom, a *AvgPool* sa bijelom/svijetlom pozadinom.

Analizom *GTSRB* skupa zaključujemo da su prisutne slike iz obje navedene "kategorije". Zbog toga nije a priori jasno koju funkciju odabrati. Nakon mnogih pokušaja pokazalo se da je neuronska mreža za ovaj problem u prosjeku točnije klasificirala uz *MaxPool*. Finalni izgled mreže zapravo se najbolje može opisati isječkom koda koji definira sve elemente mreže:

```
self.conv1 = nn.Conv2d(in_channels=3,
                       out_channels=100,
                       kernel_size=(5,5),
                       stride=1,
                       padding=2)
self.batch1 = nn.BatchNorm2d(100)
self.relu1 = nn.ReLU()
self.pool1 = nn.MaxPool2d(kernel_size=2)

# Constraints for level 2
self.conv2 = nn.Conv2d(in_channels=100,
                       out_channels=150,
                       kernel_size=(3,3),
                       stride=1,
                       padding=2)
self.batch2 = nn.BatchNorm2d(150)
self.relu2 = nn.ReLU()
self.pool2 = nn.MaxPool2d(kernel_size=2)

# Constraints for level 3
self.conv3 = nn.Conv2d(in_channels=150,
                       out_channels=250,
                       kernel_size=(3, 3),
                       stride=1,
                       padding=2)
self.batch3 = nn.BatchNorm2d(250)
self.relu3 = nn.ReLU()
self.pool3 = nn.MaxPool2d(kernel_size=2)

# Defining the Linear layer
self.fc1 = nn.Linear(250 * 5 * 5, 350)
self.fc2 = nn.Linear(350,
                     number_of_classes)
```

Korištena neuronska mreža ima tri skrivena sloja. Još je potrebno definirati pogodan broj epoha i veličinu batcha. U literaturi smo pronašli brojne zaključke o utjecaju tih veličina na točnost mreže. Jasno je da broj epoha ne smije biti premalen jer dolazi do *underfitting*-a, ali niti prevelik jer će tada model precizno aproksimirati skup za treniranje, ali će njegova generalizacija biti loša (*overfitting*). Također, veći broj epoha rezultira dužim vremenom učenja mreže pa smo za naše eksperimente uglavnom koristili $10-20$ epoha. Veličina batcha također je često diskutirana tema. Zbog arhitekture računala, veličina batcha bi trebala biti neka potencija broja 2. U našem slučaju, eksperimentiranje s različitom veličinom batcha (od 4 do 512) nije davalo značajno različite rezultate. Mala veličina batcha rezultira češćim računanjem gradijenta. Zbog toga je treniranje mreže stabilnije. Više o razlozima odabira malog batcha može se pronaći u [11]. Zbog toga odabiremo batch veličine 8 za testiranje algoritma.

V. Rezultati

A. Redukcija podataka

U prvom dijelu ovog poglavlja prezentiramo relevantne činjenice za sam proces reduciranja podataka, dok su u dijelu vezanom za CNN navedeni i rezultati ukupnog procesa. Za početak nagalasimo da se skup za treniranje sastoji od 39209 slika.

1) Značajke: Odabrane Haar-like značajke su u različitim varijantama kombinacija tipa 2 i tipa 4 ili kombinacija sva tri tipa značajki. Dodatno, proces je proveden i uz univarijatnu analizu za odabir značajki, gdje se ciljalo ukloniti 50% značajki. U sva tri slučaja podatci su reducirani s 39209 slika na 22 do 23 tisuće slika, što je očito značajna redukcija. Primjerice, u kombinaciji svih triju tipova značajki njihov ukupan broj bio je 2465, odnosno 1232 nakon provedenog (univarijatnog) odabira značajki.

2) Klasifikacija odbačenih podataka: Esencijalno eliminacijski test redukcije podataka upravo je kvaliteta klasifiranja odbačenih podataka uz pomoć zadržanih podataka. Ukoliko s odabranih K slika ne možemo klasificirati ostalih $N-K$ slika, kompletan proces i početne hipoteze o redukciji propadaju. No, u velikoj većini testova provedenih u sklopu ovog rada, točnost klasifikacije kada se za treniranje koristi 23000 slika odabranih kroz klasteriranje, a za testiranje koristi ostalih ≈ 16000 slika jest veća od 99% pa čak i veća od 99.5%. Povremeno se kroz testove postigla točnost nešto niža od 99%, što je bio dobar indikator da odabrani parametri za redukciju podataka nisu optimalni. Dakle, ovi su rezultati u neku ruku potvrdili našu početnu hipotezu i učinili osnovno pitanje validnim - *Je li svih* 39000 *slika nužno za dobru klasifikaciju?*

3) Vrijeme izvođenja: U hipotezama projektnog prijedloga cilj je bio da ukupno trajanje redukcije podataka bude reda veličine najviše jednog treniranja konvolucijske neuronske mreže. Konkretno, u ovoj izvedbi redukcija podataka s 2465 značajki trajala je ukupno ≈ 40 minuta, dok je varijanta s odabirom značajki trajala ≈ 30 minuta. S obzirom da su se vremena treniranja neuronske mreže kroz sve provedene testove kretala uglavnom u rangu $30-150$ minuta, zaključujemo da je ukupno vrijeme potrebno za redukciju podataka ispunilo očekivanja.

B. Konvolucijska neuronska mreža

U velikoj većini slučajeva testiranja konvolucijske neuronske mreže točnost klasifikacije bila je u intervalu $93\% - 96\%$. U tablici ispod navodimo nekoliko primjera rezultata klasifikacije za različite, dosad navedene izbore parametara.

Redukcija	Odabir značajki	Trajanje	Točnost
Ne	Ne	133 min	95.91%
Da	Ne	65 min	96.28%
Da	Da	46 min	94.95%

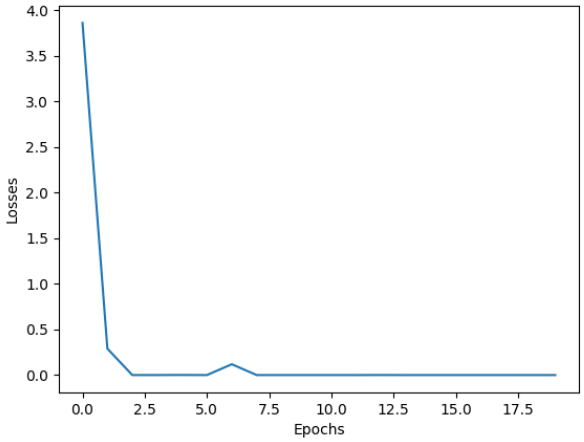
Osim naših rezultata, navodimo rezultate originalnog GTSRB natjecanja. (Slika 10)

TEAM	METHOD	TOTAL
[156] DeepKnowledge Seville	CNN with 3 Spatial Transformers	99.71%
[3] IDSIA	Committee of CNNs	99.46%
[155] COSFIRE	Color-blob-based COSFIRE filters for object recogn	98.97%
[1] INI-RTCV	Human Performance	98.84%
[4] sermanet	Multi-Scale CNNs	98.31%
[2] CAOR	Random Forests	96.14%
[6] INI-RTCV	LDA on HOG 2	95.68%
[5] INI-RTCV	LDA on HOG 1	93.18%
[7] INI-RTCV	LDA on HOG 3	92.34%

Slika 10. Rezultati GTSRB natjecanja

Jasno je da točnost opisane neuronske mreže nije idealna. Međutim, svakako je konkurentna budući da se njena točnost nalazi pri samom vrhu ondašnje tablice. Do danas su razvijene mnoge druge arhitekture mreže koje daju puno bolju točnost te se mogu pronaći u [2].

Zanimljiva je primjedba da je točnost mreže uz jednake parametre sa i bez data reductiona uvijek bila približno jednaka. U nekim slučajevima je bila bolja prva, a u nekima druga opcija.



Slika 11. Loss funkcija uz data reduction

VI. Zaključak

Proces redukcije podataka u smislu poboljšavanja rada konvolucijske neuronske mreže pokazao se uspješnim. Osim što se vrijeme treniranja značajno smanjilo, točnost je ostala praktički jednaka, a katkad čak i bolja, što znači da su esencijalno ostvareni glavni ciljevi rada.

Idući koraci za poboljšavanje ovog algoritma definitivno uključuju izbor dodatnih značajki povrh korištenja isključivo Haar-like značajki i shodno tome metodičniji pristup odabiru najboljih značajki. Dodatno, moguće je dodatno poboljšati neuronsku mrežu korištenjem naprednijih koncepata poput *spatial transformer*-a te provjeriti hoće li se data reduction pristup i tada uspjeti održati na adekvatnoj razini.

Sve u svemu, zaključujemo da je ovakav pristup osnovan i smislen. Osim u svrhu brže i bolje klasifikacije, još jedna primjena data reductiona mogla bi biti olakšano testiranje novih parametara ili posljedica uvođenja novih podataka u mrežu. Stoga smatramo da se apsolutno isplati nastaviti provoditi istraživanje u smjeru poboljšavanja ovog koncepta.

Literatura

[1] Vighnesh Birodkar Hossein Mobahi Samy Bengio: *Semantic Redundancies in Image-Classification Datasets: The 10% You Don't Need*, 2019.
[2] https://benchmark.ini.rub.de/ *GTSRB dataset*
[3] Verónica Bolón-Canedo, Beatriz Remeseiro, *Feature selection in image analysis: a survey*, 2019.
[4] Paul Viola, Michael Jones: *Rapid Object Detection using a Boosted Cascade of Simple Features*, 2001.
[5] Miron B. Kursa,Witold R. Rudnicki: *Feature Selection with the Boruta Package*, JSS Journal of Statistical Software, 2010.
[6] Yanchi Liu1, Zhongmou Li, Hui Xiong, Xuedong Gao, Junjie Wu: *Understanding of Internal Clustering Validation Measures*, 2010 IEEE International Conference on Data Mining
[7] Maria Halkidi, Michalis Vazirgiannis: *Clustering Validity Assessment: Finding the optimal partitioning of a data set*, 2001.
[8] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander: *OPTICS: Ordering Points To Identify the Clustering Structure*, 1999.
[9] *https://iq.opengenus.org/maxpool-vs-avgpool/*
[10] Pedregosa et al.: *Scikit-learn: Machine Learning in Python*, 2011.
[11] Dominic Masters, Carlo Luschi: *Revisiting Small Batch Training for Deep Neural Networks*, 2018.