SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2287

INTERPRETIRANJE DISKRIMINATIVNIH KONVOLUCIJSKIH MODELA

Tin Ceraj

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2287

INTERPRETIRANJE DISKRIMINATIVNIH KONVOLUCIJSKIH MODELA

Tin Ceraj

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

DIPLOMSKI ZADATAK br. 2287

Pristupnik:	**Tin Ceraj (0036485822)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Interpretiranje diskriminativnih konvolucijskih modela**

Opis zadatka:

Duboki konvolucijski modeli su glavni sastojak mnogih praktičnih primjena računalnog vida. Međutim, često se javlja kritika da takvi modeli nisu spremni za industrijsku upotrebu zbog loše interpretabilnosti, odnosno, nemogućnosti modela da ono što je naučio prikaže ljudima. Zbog toga postupci za interpretiranje konvolucijskih modela predstavljaju zanimljivo područje istraživanja. U okviru rada, potrebno je proučiti i reproducirati metode za interpretiranje konvolucijskih arhitektura za klasifikaciju slike. Oblikovati postupak za uzorkovanje slika naučenog diskriminativnog modela. Validirati hiperparametre, prikazati i ocijeniti ostvarene rezultate te provesti usporedbu s rezultatima iz literature. Predložiti pravce budućeg razvoja. Radu priložiti izvorni kod razvijenih postupaka uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 30. lipnja 2020.

Zahvaljujem svom mentoru prof. dr. sc. Siniši Šegviću na pomoći i savjetima koje mi je pružio tijekom diplomskog studija.

SADRŽAJ

1. Uvod

Sa sve većom upotrebom umjetne inteligencije i računalnog vida u stvarnom svijetu, pojavljuju se novi problemi. U nekim granama industrije nije dovoljno da model može zaključiti nešto na temelju ulaznih informacija, već je potrebno pokazati da je taj zaključak valjan i od kuda proizlazi.

Kod dubokih konvolucijskih modela taj problem je još veći jer su ti modeli često korišteni kao crne kutije, bez ikakvog znanja o tome kako rade. Očekivano je da se često javlja kritika da takvi modeli nisu spremni za industrijsku upotrebu zbog loše interpretabilnosti, odnosno, nemogućnosti modela da ono što je naučio prikaže ljudima. Kako bi se pokazalo da model zaista radi na čovjeku smislen način, postoje metode vizualizacije i objašnjavanja modela.

Neke od tih metoda su metoda dubokog sna (engl. *DeepDream*) (Mordvintsev et al., 2015) te metoda duboke inverzije (engl. *DeepInversion*) (Yin et al., 2020) koja generira slike tražene oznake koristeći naučeni model. Ciljani rezultat generiranja je kvaliteta i realnost slike kako bi se dokazalo znanje modela.

Prednost ovih metoda je što ne zahtijevaju nikakve ulazne podatke. Kao ulaz primaju nasumične slike, odnosno šum i iterativno ih optimiziraju u tražene klase. Vode se gubitkom unakrsne entropije kombiniranim sa skupom gubitaka regularizacije s ciljem generiranja slika sličnim onima iz skupa za učenje koje je model koristio tijekom treniranja.

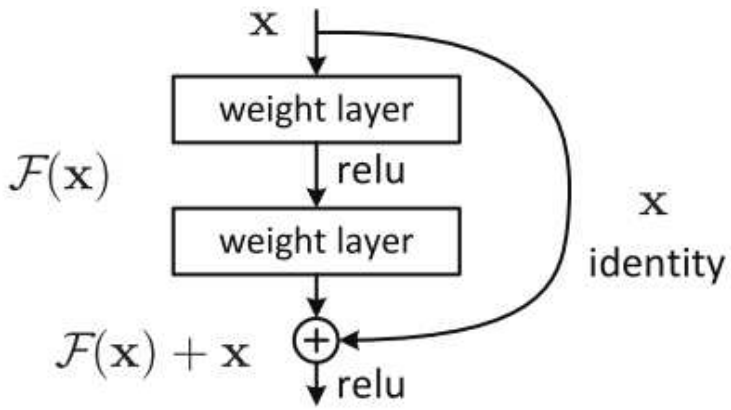
Ovaj rad bazira se na navedenim metodama te ih implementira i testira sa različitim hiperparametrima i uvjetima.

2. Duboki modeli

Najčešći modeli korišteni u računalnom vidu su duboki modeli. Konvolucijski modeli pokazuju najbolje rezultate i u pravilu se oni i koriste. U ovom radu će naglasak biti na arhitekturi Rezidualne neuronske mreže (engl. *Residual Neural Network*) tj. *ResNet* (He et al., 2016) arhitekturi, jednoj od najpoznatijih konvolucijskih arhitektura.

2.1. Arhitektura Resnet

Ideja za arhitekturu rezidualne mreže motivirana je problemom degradacije koji se pojavljuje u dubokim modelima i raste s dubinom modela, a uzrokuju ga nestajući gradijenti. Autori Resnet-a uvode koncept rezidualnih veza (He et al., 2016) koje spajaju ulaze u skup slojeva na izlaz tog skupa. Pod pretpostavkom da neki skup slojeva mapira ulaz **x** na izlaz $\mathcal{H}(\mathbf{x})$, onda taj isti skup može mapirati taj isti ulaz na $\mathcal{H}(\mathbf{x})$ - **x**. Sad je umjesto mapiranja na $\mathcal{H}(\mathbf{x})$ izlaz rezidualne funkcije $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x})$ - **x** što znači da je funkcija mapiranja koju slojevi uče jednaka $\mathcal{F}(\mathbf{x})$ + **x**.

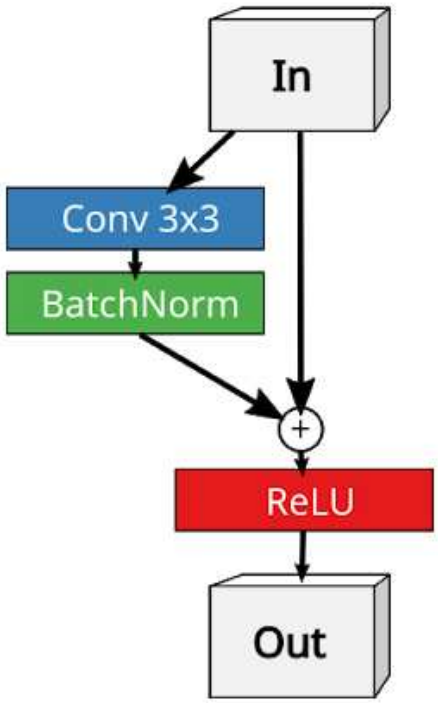


Slika 2.1: Osnovni ResNet blok

Kao što je prikazano na slici 2.1, ulaz u ResNet blok je spojen na njegov izlaz. Time je postignuto opisano mapiranje s ulaza **x** na izlaz $\mathcal{F}(\mathbf{x})$ + **x**. Glavna ideja ove

veze je direktno preslikavanje funkcije identiteta odnosno da izlaz bloka bude jednak njegovom ulazu. U slučaju da je to potrebno, $\mathcal{F}(\mathbf{x})$ će biti postavljen na 0 i time će izlaz biti samo $\mathbf{x}$.

Najčešći slojevi u ResNet-u su konvolucijski, slojevi normalizacije grupa (engl. *batchnorm*) i ReLU slojevi, a na početku i kraju su prisutni i MaxPool, AvgPool te potpuno povezani slojevi. Mreža se sastoji od blokova u kojima iza svakog konvolucijskog sloja slijedi sloj normalizacije grupa, a blok završava ReLU slojem kao što je prikazano na slici 2.2.



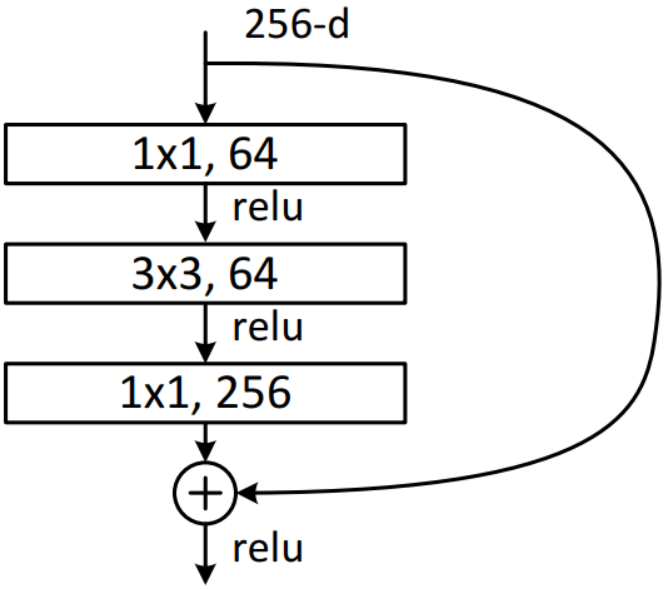
Slika 2.2: Osnovni ResNet blok

Blokovi sadrže proizvoljni broj parova konvolucijskih slojeva i slojeva normalizacije grupa nakon čega zajedno s ulazom u blok prolaze kroz ReLU sloj na kraju bloka.

2.2. ResNet50

ResNet50 baziran je na arhitekturi ResNet i sastoji se od pedeset slojeva. Na početku se nalazi jedan par konvolucijskog sloja i sloja normalizacije grupa nakon čega slijede ReLU i MaxPool sloj dok se na izlazu nalaze *AdaptiveAvgPool* i potpuno povezani sloj. U sredini je glavni dio mreže koji se sastoji od rezidualnih gradivnih blokova, koji se ponavljaju šesnaest puta, a sastoje se od tri para konvolucijskih slojeva i slojeva

normalizacije grupa s ReLU slojem na kraju kao na slici 2.3. Konvolucijski filteri u tim blokovima imaju veličinu filtera 1x1, 3x3, 1x1, redom.



Slika 2.3: ResNet *bottleneck* blok

2.3. Robusni modeli

Robusnost je u kontekstu dubokog učenja definirana kao sposobnost modela da dobro radi u uvjetima kada je okruženje drugačije od onog na kojem je treniran. Jedan od načina postizanja robusnosti je učenje s neprijateljskim primjerima. U ovom radu se koristi model iz paketa robustness Engstrom et al. (2019) treniranog na skupu podataka ImageNet s vrijednosti $\epsilon = 8/255$.

Učenje modela na način da se generiraju neprijateljski primjeri sa svrhom da model postane otporan na njih u pravilu rezultira robusnim modelom otpornim na neprijateljske primjere iz domene učenja. Samo učenje može se opisati kao optimizacijski problem (Božić, 2019):

$$\min_{\theta} \frac{1}{|D_{train}|} \sum_{(x,y)\in D_{train}} \max_{\delta\in\Delta(x)} l(h_\theta(x+\delta), y) \tag{2.1}$$

što je zapravo minimizacija sume gubitaka koji su maksimizirani s odabranom perturbacijom za svaki primjer.

Optimizacijski problem naveden u (2.1) se onda rješava optimiziranjem parametra θ tj. parametara modela kroz mini-grupe $B \subseteq D_{train}$ sa sljedećom formulom:

$$\theta := \theta - \frac{\alpha}{|B|} \sum_{(x,y)\in B} \nabla_\theta \max_{\delta\in\Delta(x)} l(h_\theta(x+\delta), y) \tag{2.2}$$

Sastavni dio izraza (2.2) je gradijent unutar kojeg se nalazi optimizacijski problem. Takav problem se rješava Danskinovim teoremom (Madry et al., 2018).

3. Vizualizacija dubokih modela

Kako bi se čovjeku moglo prikazati što je neki model naučio, potrebno je model na neki način vizualizirati. U radu se vizualiziraju diskriminativni konvolucijski modeli koji se koriste za klasifikaciju slika. Modele nije potrebno mijenjati već samo pokrenuti s nekom od metoda za vizualizaciju.

Modeli navedeni u prethodnom poglavlju vizualizirani su korištenjem metoda za vizualizaciju poput DeepDream (Mordvintsev et al., 2015) i DeepInversion (Yin et al., 2020). U radu je naglasak na metodi DeepInversion s obzirom da je metoda zapravo nadogradnja na DeepDream i daje bolje tj. realnije rezultate.

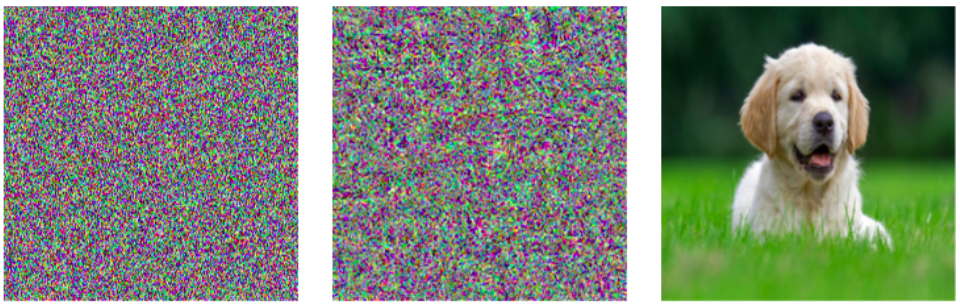
Općenito, ove metode su bazirane na minimiziranju funkcije gubitka i računanju gradijenta ulaznih parametara, odnosno slike. Najjednostavniji oblik te minimizacije može se dobiti definiranjem gubitka s obzirom na jednu od klasa koje model može klasificirati. Uzme se npr. gubitak unakrsne entropije (engl. *cross-entropy loss*) i minimizira se s obzirom na ulaz $\hat{x}$ za odabranu klasu y, formula (3.1). Kao početni ulaz, odabran je skup nasumičnih piksela tj. šum (engl. *noise*). Unazadnom propagacijom (engl. *backpropagation*) dobijaju se gradijenti ulazne slike i slika se pomiče u smjeru gradijenata.

$$\min_{\hat{x}} = \mathcal{L}(\hat{x}, y) \tag{3.1}$$

Kroz samo nekoliko iteracija, slika se sa sve većom sigurnošću klasificira kao odabrana klasa. Problem kod ove metode je što se ulazna slika prilagodila modelu tj. njegovoj klasifikaciji u odabranu klasu bezuvjetno tj. bez ikakve regularizacije. S obzirom da se aktivacije povećavaju s apsolutnim vrijednostima piksela i promjenama susjednih piksela tj. uzorcima, rezultat toga su nerealne slike koje maksimiziraju aktivacije s ciljem minimizacije funkcije gubitka za željenu predikciju.

Primjer optimizacije se može vidjeti na slici 3.1. Na lijevoj slici je nasumični šum, dok je na srednjon slici taj šum optimiziran gradijentnim spustom u klasu s oznakom 207, odnosno "zlatni retriver". Srednja slika, iako ljudskom oku ne sliči na zlatnog retrivera, sadrži uzorke koji aktiviraju pojedine neurone u mreži i maksimiziraju izlaz

za klasu 207. Desna slika prikazuje zlatnog retrivera, ali ga mreža klasificira s manjom sigurnosti od srednje slike. To se događa jer je srednja slika dobijena korišenjem znanja o modelu i iskorištavanjem jakih aktivacija. Zbog toga se u ovakve optimizacije uvode regularizacije jer one ograničavaju mogućnost iskorištavanja ekstremnih aktivacija. Uz regularizacije se također uvode i neke druge metode koje npr. ograničavaju vrijednost piksela.



Slika 3.1: Lijeva slika prikazuje nasumični šum; srednja slika prikazuje taj isti šum optimiziran u zlatnog retrivera; desna slika prikazuje zlatnog retrivera

Kako bi se izbjegli rezultati dobiveni optimizacijom isključivo unakrsne entropije, u optimizaciju je potrebno uključiti regularizaciju koja će odbijati algoritam od direktne minimizacije.

Metode *DeepDream* i *DeepInversion* primjenjuju nekoliko različitih regularizacija od kojih svaka ima svoju svrhu i na neki način poboljšava kvalitetu i realnost generiranih slika.

3.1. DeepDream

Implementiran 2015. godine, *DeepDream* je osmišljen da uzorkuje umjetničke elemente iz slika. Također se može koristiti za optimiziranje šuma u sliku. Autori *DeepDream*-a predstavili su regularizaciju za nasumični ulaz $\hat{x} \in \mathcal{R}^{H \times W \times C}$, gdje su H, W, C visina, širina i broj kanala boje i odabrana klasa y, slika se uzorkuje optimiziranjem

$$\min_{\hat{x}} = \mathcal{L}(\hat{x}, y) + \mathcal{R}(\hat{x}), \tag{3.2}$$

gdje je $\mathcal{L}(\cdot)$ gubitak unakrsne entropije, a $\mathcal{R}(\cdot)$ regularizacijski izraz (Yin et al., 2020).

U *DeepDream*-u se koristi *image prior* regularizacijski izraz kako bi uzorkovane slike bile realnije:

$$\mathcal{R}_{prior}(\hat{x}) = \alpha_{tv}\mathcal{R}_{TV}(\hat{x}) + \alpha_{l2}\mathcal{R}_{l2}(\hat{x}), \tag{3.3}$$

gdje su R_{TV} i R_{l2} regularizacijski izrazi koji kažnjavaju totalnu varijaciju i L2 normu ulaza $\hat{x}$, dok faktori α_{tv}, α_{l2} služe za mijenjanje snage regularizacijskih izraza. Yin et al. (2020) navode da je empirijskim promatranjem pokazano da korištenje *image prior* izraza (3.3) doprinosi stabilnoj konvergenciji prema realnijim slikama. Iako je to korak naprijed, slike generirane ovom metodom još uvijek nisu dovoljno realne.

Slika 3.2: Slike generirane DeepDream-om

3.2. DeepInversion

DeepInversion, povrh *image prior* izraza, dodaje nove regularizacijske izraze s ciljem uzorkovanja realnijih slika. Autori rada (Yin et al., 2020) tvrde da sami *image prior* izraz ne usmjerava uzorkovanje dovoljno da bi generirana slika $\hat{x} \in \hat{\mathcal{X}}$ imala slične značajke kao prave slike $x \in \mathcal{X}$. Uz to, da bi značajke slike imale sličnosti na svim razinama složenosti s onima na pravim slikama, potrebno je minimizirati udaljenost između statistika značajki generiranih slika $\hat{x}$ i pravih slika x.

Pretpostavljaju da statistike značajki kroz grupu prate Gaussovu distribuciju i da se zbog toga mogu definirati aritmetička sredina i varijanca grupe. Uz tu pretpostavku, regularizacijski izraz za distribuciju značajki se može definirati kao:

$$\mathcal{R}_{feature}(\hat{x}) = \sum_{l} \|\mu_l(\hat{x}) - \mathbb{E}(\mu_l(x)|\mathcal{X})\|_2 + \sum_{l} \left\|\sigma_l^2(\hat{x}) - \mathbb{E}(\sigma_l^2(x)|\mathcal{X})\right\|_2 \quad (3.4)$$

gdje su $\mu_l(\hat{x})$ i $\sigma_l^2(\hat{x})$ aritmetička sredina i varijanca grupe na l-tom konvolucijskom sloju, redom. Operatori $\mathbb{E}(\cdot)$ i $\|\cdot\|$ su definirani kao očekivana vrijednost i $l2$ norma, redom.

S obzirom da za računanje očekivanja aritmetičke sredine i varijance tj. $\mathbb{E}(\mu_l(x)|\mathcal{X})$ i $\mathbb{E}(\sigma_l^2(x)|\mathcal{X})$, potreban skup za učenje, Yin et al. (2020) aproksimiraju njihove vrijednosti pomoću prosječnih statistika spremljenih u *BatchNorm* slojeve. *BatchNorm* sloj nalazi se nakon svakog konvolucijskog sloja i normalizira mape značajki tijekom treniranja kako bi se izbjegao kovarijacijski pomak. Također implicitno pamti aritmetičku sredinu i varijancu po kanalu što omogućava aproksimaciju jednadžbe (3.4) sljedećim izrazom:

$$\mathbb{E}(\mu_l(x)|\mathcal{X}) \simeq \text{BN}_l(\text{running\_mean}) \quad (3.5)$$

$$\mathbb{E}(\sigma_l^2(x)|\mathcal{X}) \simeq \text{BN}_l(\text{running\_variance}) \quad (3.6)$$

Autori su pokazali da navedeni regularizacijski izraz za distribuciju značajki (3.4) značajno poboljšava kvalitetu generiranih slika. Konačni izraz metode *DeepInversion* je:

$$\mathcal{R}_{\text{DI}}(\hat{x}) = \mathcal{R}_{prior}(\hat{x}) + \alpha_f \mathcal{R}_{feature}(\hat{x}) \quad (3.7)$$

gdje su α_f i $\mathcal{R}_{feature}(\hat{x})$ definirani kao koeficijent snage regularizacije distribucije značajki i sama regularizacija distribucije značajki, redom.

DeepInversion daje realistične slike koje imaju odabranu klasu generiranu kao cjelinu umjesto kao skup nekih značajki. Slike generirane iz rada Yin et al. (2020) se mogu vidjeti na slici 3.3. Sve klase sa slike su ljudskom oku prepoznatljive.

3.3. Dodatne metode

Uz navedene regularizacije, korištene su i neke pomoćne metode za obradu slika s ciljem poboljšanja realnosti generiranih slika.

Podrezivanje slika (engl. *image clipping*) ograničava minimalnu i maksimalnu vrijednost boja na pikselima. Može se implementirati fiksno ili varijabilno ograničenje.

Slika 3.3: Slike generirane DeepInversion-om

U radu se koristi varijabilno ograničenje koje je opisano u Yin et al. (2020). U svakoj iteraciji se podrezuju vrijednosti u odnosu na aritmetičku sredinu i standardnu devijaciju. Metoda poboljšava realnost slika jer smanjuje količinu neprirodnih vrijednosti boja.

Nasumično zrcaljenje i rotacija piksela također pomažu kod generacije realističnih slika. Glavna prednost je brža konvergencija u realističe slike u situacijama kad su određeni pikseli "zapeli" ili kad je model naučio neke asimetrične značajke.

U početku se koristio jednostavni gradijentni spust, ali se ubrzo zamijenio s Adam optimizatorom (Kingma i Ba, 2014) jer se pokazao kao najbolji za generiranje slika.

Uz navedeno, pri generaciji slika se dimenzije slike te veličina grupe mogu proizvoljno podesiti. Jedino ograničenje je količina grafičke memorije. Metoda koja pomaže pri tome je višedimenzionalno generiranje slika (engl. *Multi-resolution syn-*

thesis) tj. provlačenje slike kroz AvgPool sloj koji smanjuje dimenzije prije ulaska u model. Time se značajno smanjuje količina korištene memorije i vrijeme izvođenja.

4. Eksperimenti

U ovom poglavlju, prikazani su pokušaji generiranja slika i njihovi rezultati uz opise postupaka kojima su ti rezultati dobiveni. Kao što se može vidjeti, svaki dodani izraz regularizacije na neki način doprinosi realnosti slike. Glavni cilj bio je vizualizirati znanje modela, odnosno vidjeti što je model naučio.

Osim regularizacije, u eksperimentima se koriste dodatne metode za obradu slike navedene u odjeljku 3.3. One doprinose brzini izvođenja i pomažu regularizacijskim izrazima kod ostvarivanja realnosti na slikama. Višedimenzionalno generiranje se primjenjuje u svim eksperimenata zbog brzine izvođenja, a ostale metode, ako nije drugačije navedeno, su primjenjene u svim eksperimentima.

4.1. Regularizacijski izrazi

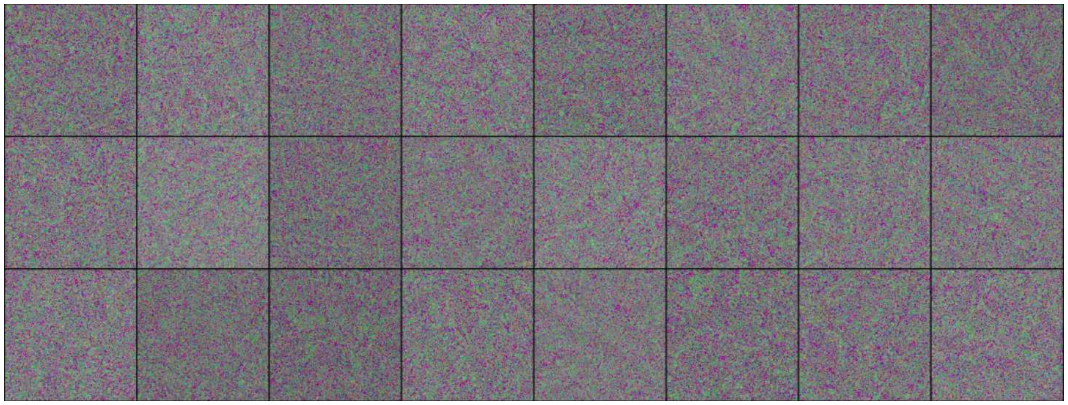
U ovoj sekciji prikazani su rezultati dodavanja regularizacijskih izraza i napredak koji su isti izrazi ostvarili u generaciji slika. Koristi se Adam optimizator s vrijednostima β $0.9, 0.999$ i ϵ 10^{-8}, stopa učenja je 0.05.

4.1.1. Minimizacija klasifikacijskog gubitka

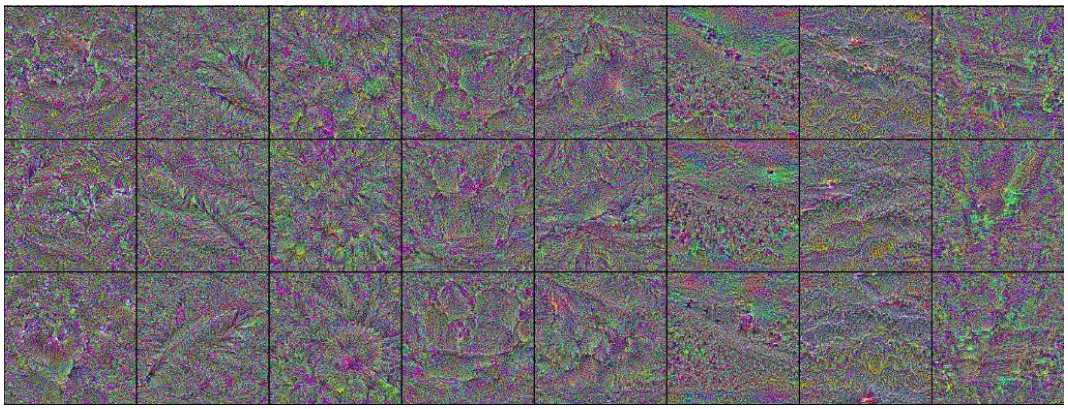
Prije dodavanja bilo kakve regularizacije, sve što je minimizirano je klasifikacijski gubitak, a korišten je gubitak unakrsne entropije (3.1). Dobijene slike su klasificirane kao željena klasa i uzorci se na njima vide, ali i dalje više izgledaju kao šum nego kao realna slika.

Kao što se vidi na slici 4.1, generirane slike nisu dovoljno realne. Na njima se vide uzorci tj. pravilnosti. Sve slike su klasificirane u odabrane klase sa približno 100% sigurnosti. Na slici 4.2 se koriste dodatne metode za obradu slika. Ni te slike nisu realne, ali se ipak mogu vidjeti uzorci bolje nego na slikama generiranima bez korištenja dodatnih metoda.

Kao što se može vidjeti na slikama 4.1 i 4.2, korištenjem navedenih metoda se



Slika 4.1: Slike generirane bez regularizacije



Slika 4.2: Slike generirane bez regularizacije uz korištenje *image clipping*-a, nasumičnog zrcaljenja i rotacije

značajke bolje vide. Najveći doprinos radi podrezivanje slika jer podrezuje koje jako odstupaju od srednjih vrijednosti i rijetko se pojavljuju na slikama. Ostale metode ne pridonose vizualno koliko pomažu da algoritam brže konvergira.

4.1.2. Image prior izraz

Image prior izraz sastoji se od dvije komponente, totalne varijacije ulaza i L2 norme ulaza. Totalna varijacija računa se sa sljedećom formulom

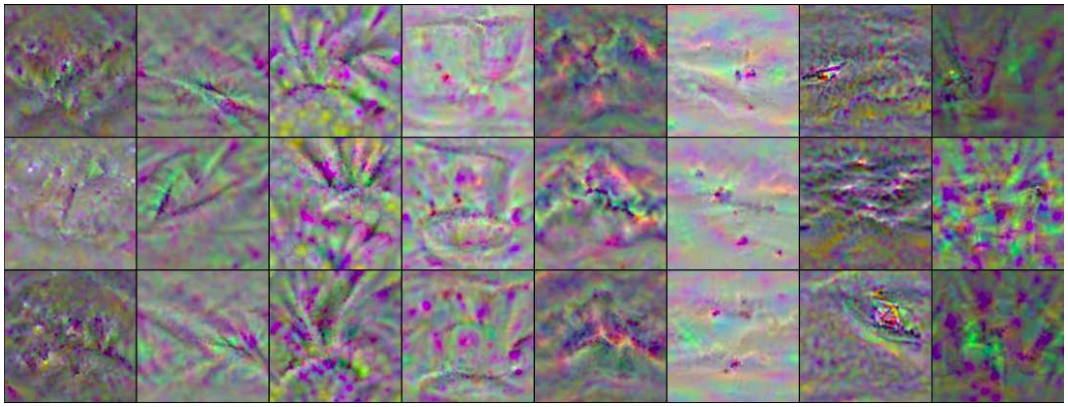
$$\mathcal{R}_{TV}(\hat{x}) = \sqrt{|y_{i+i,j} - y_{i,j}| + |y_{i,j+1} - y_{i,j}|} \tag{4.1}$$

Uzevši u obzir da ograničavanje totalne varijacije smanjuje količinu naglih promjena, smanjuje se šum na slici jer je manja vjerojatnost da će susjedni pikseli jako odstupati jedni od drugih. To ne znači da nikada neće odstupati, već samo da će svako odstupanje biti opravdano jer će u tom slučaju aktivacija za neku značajku biti jača od

gubitka totalne varijacije kojeg to odstupanje uzrokuje.

Regularizacija slike L2 normom kažnjava velike vrijednosti boja pa će slika imati manje piksela s ekstremnim vrijednostima boja. Nije toliko značajna kao gubitak totalne varijacije, ali ipak pomaže kod dodavanja realnosti jer ekstremne vrijednosti oku izgledaju neprirodno.

Pri pokretanju koda koji je generirao slike sa slike 4.3, vrijednosti skaliranja regularizacija su α_{l2} 10^{-5}, α_{tv} 10^{-3}.



Slika 4.3: Slike generirane s *image prior* regularizacijom. Klase na slikama su redom po stupcima s lijeva na desno: cheeseburger, pero (engl. *quill*), tratinčica (engl. *daisy*), šalica (engl. *cup*), vulkan (engl. *vulcano*), plaža (engl. *seaside*), gliser (engl. *speedboat*) i saksofon (engl. *saxophone*)

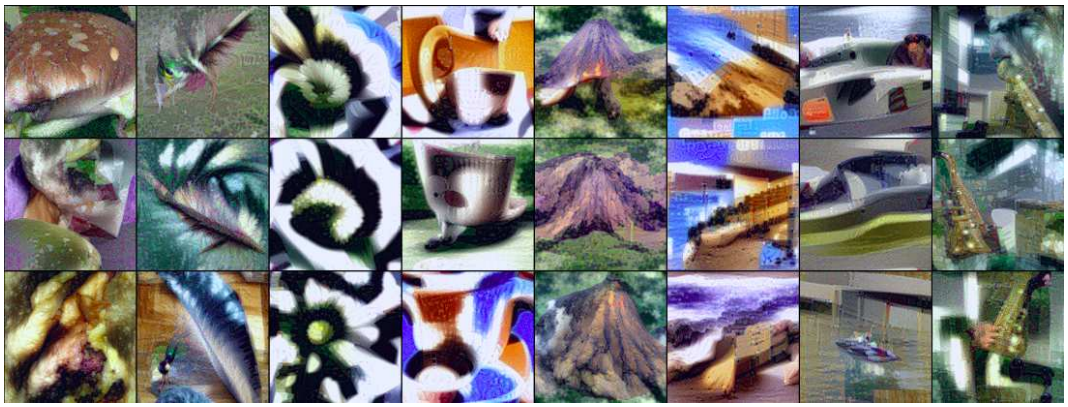
Na ovim slikama se vidi utjecaj gubitka *image prior* izraza u obliku zamućenja i značajno manjeg šuma. Na nekim klasama se mogu primijetiti oblici, npr. na klasi cheeseburger (prvi stupac s lijeve strane) se može vidjeti oblik peciva na gornjoj i donjoj slici, dok su na sredini slika neki uzorci koji vjerojatno potječu od mapiranja sastojaka cheeseburgera. Na klasi gliser (drugi stupac s desne strane) se mogu vidjeti uzorci valova i na jednom dijelu slike nedostatak tih uzoraka što je vjerojatno pozicija glisera na slici.

Ovaj eksperiment je baza za *DeepDream* i s malo promjena mogu se generirati slike slične onima iz rada (Mordvintsev et al., 2015).

4.1.3. Statistike batchnorm slojeva

Glavna pretpostavka *DeepInversion*-a je da statistike značajki generirane slike trebaju imati slične vrijednosti pravim slikama te da grupa generiranih slika može dovoljno dobro aproksimirati vrijednosti pravih slika.

Korištena veličina grupe je 84, veličina slika je 112x112 2000 iteracija pa 224x224 sljedećih 1000, a vrijednost skaliranja regularizacije značajki grupa $\alpha_f = 0.01$. Totalna varijacija i L2 norma su iste kao i u eksperimentu s rezultatima na slici 4.3 te su njihove skalarne vrijednosti također iste. Regularizacija značajki se računa kako je opisano u izrazu (3.4) i provodi se nakon svake iteracije.



Slika 4.4: Slike generirane uz implementaciju regularizacije statistike značajki grupa. Klase na slikama su redom po stupcima s lijeva na desno: cheeseburger, pero (engl. *quill*), tratinčica (engl. *daisy*), šalica (engl. *cup*), vulkan (engl. *vulcano*), plaža (engl. *seaside*), gliser (engl. *speedboat*) i saksofon (engl. *saxophone*)

Slike generirane ovom metodom su značajno realnije i na slikama se razaznaju detalji koje i ljudsko oko prepoznaje. Na cheeseburgeru se jasno vidi pecivo na prvoj slici, pero se donekle može razaznati, tratinčica ima značajke cvijeta osim što je krive boje, vulkan ima sve značajke vulkana, šalica i saksofon počinju poprimati oblik, gliser na zadnjoj slici izgleda kao da se nalazi u moru, dok su slike plaže manje uspješne.

Treba primjetiti da su boje na slikama najveći napredak. Može se reći da normalizacija statistika značajki grupa radi i da dobro utječe na boje. Iz ovih slika se vidi da bolje radi na predmetima koji imaju neke specifične značajke poput saksofona ili cheeseburgera, a čak i na lošem primjeru poput plaže su boje u dobrom odnosu - plavo more na jednoj strani slika, a obala na drugoj.

4.2. Robusni modeli

Robusni modeli korišteni u radu uzeti su iz paketa robustness (Engstrom et al., 2019). Pretrenirani ImageNet model treniran je s perturbacijama $\epsilon\ 8/255$. Kod generacije slika, robusni modeli daju puno realnije slike na kojima je ljudskom oku odmah jasno o kojoj se klasi radi.

S obzirom da klasifikacija robusnog modela ne smije ovisiti o malim značajkama, koje se lako mogu izmijeniti perturbacijom, model nauči gledati "širu sliku". To znači da mu nije dovoljno da slika ima neke značajke koje se pojavljuju na odabranoj klasi, već mora imati i cjelinu kako bi aktivacija bila zadovoljavajuće snage. Rezultati se mogu vidjeti na sljedećoj slici:

Slika 4.5: Slike generirane uz implementaciju regularizacije statistike značajki grupa i korištenje robusnog modela. Klase na slikama su redom po stupcima s lijeva na desno: cheeseburger, pero (engl. *quill*), tratinčica (engl. *daisy*), šalica (engl. *cup*), vulkan (engl. *vulcano*), plaža (engl. *seaside*), gliser (engl. *speedboat*) i saksofon (engl. *saxophone*)

Generirane slike su značajno realnije od onih generiranima s nerobusnim modelima. Na generiranim slikama koje pripadaju istoj klasi se vidi sličnost, odnosno izgleda kao da model ima generalnu ideju kako neka klasa izgleda i na temelju toga generira sliku. Iz ovog se može zaključiti da model ne pamti naučene slike, već da svaku za svaku sliku zna što treba imati kako bi bila klasificirana kao odabrana klasa npr. cheeseburger mora imati gornje pecivo i punjenje. Osim prikazanih klasa, još neke klase dobro demonstriraju navedenu tvrdnju 4.6.

I na ovim slikama se dobro razaznaje kojoj klasi pripadaju. Medvjed uvijek ima njušku, oči i tijelo, jagoda ima pravilne uzorke i otprilike je dobrog oblika, mobitel ima ekran i tipkovnicu, boce imaju grlo i tijelo te naljepnicu. S druge strane, neke klase su problematične npr. baklja i luk, vjerojatno jer su u većini slučajeva kombinirane s ljudima. Na svakoj slici s bakljama se pojavljuju ljudi, a na slikama luka se vidi strijelac ili njegova ruka. Osim toga, baklja i luk se mogu vidjeti na svakoj slici.

Slika 4.6: Slike generirane uz implementaciju regularizacije statistike značajki grupa i korištenje robusnog modela. Klase na slikama su redom po stupcima s lijeva na desno: pivska boca (engl. *beer bottle*), mobitel (engl. *cell*), jagoda (engl. *strawberry*), baklja (engl. *torch*), luk (engl. *bow*), smeđi medvjed (engl. *brown bear*)

4.3. Validacija hiperparametara

Prethodni eksperimenti su koristili iste hiperparametre, a u ovom odjeljku se eksperimentira s različitim hiperparametrima. Parametri koji utječu na slike su faktori skaliranja regularizacija α_{tv}, α_{l2} i α_f i β vrijednosti Adam optimizatora.

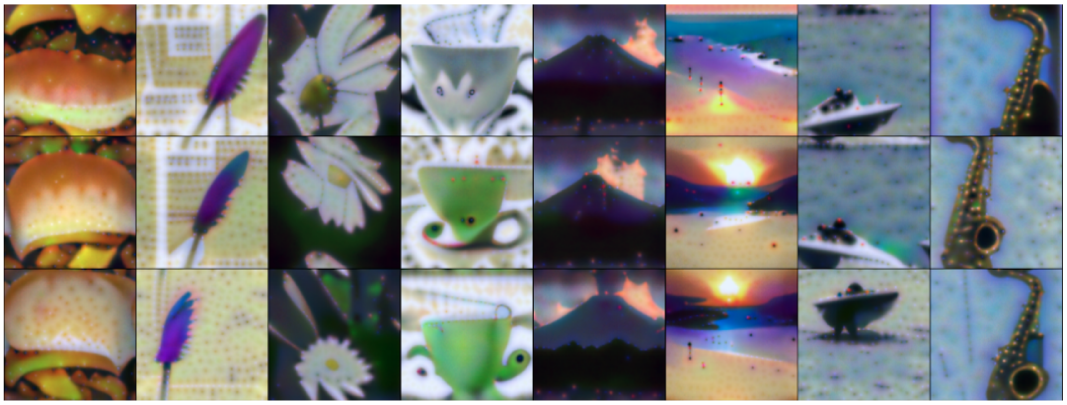
Smanjivanjem β vrijednosti Adam optimizatora radi drugačije za svaku klasu. Neke klase su značajno bolje, dok su neke lošije. Rezultati se mogu vidjeti na slikama 4.7 i 4.8. Osim što neke klase imaju bolju formu i prepoznatljive su, sve slike su mutnije od onih iz prethodnih eksperimenata.

Osim hiperparametara Adam optimizatora, drugačije slike se mogu dobiti korištenjem veće rezolucije slika. Problem kod povećavanja rezolucije slika je kvadratno povećanje potrošnje memorije grafičke kartice pa su u eksperimentu korištene dimenzije 336x336. Neke slike su poboljšanje, ali zanimljivo je primijetiti da se broj jedinki povećao na slikama. Više tratinčica, više jagoda, a čak se na jednoj slici pojavljuju dva vulkana. Rezultati se mogu vidjeti na slikama 4.9 i 4.10.

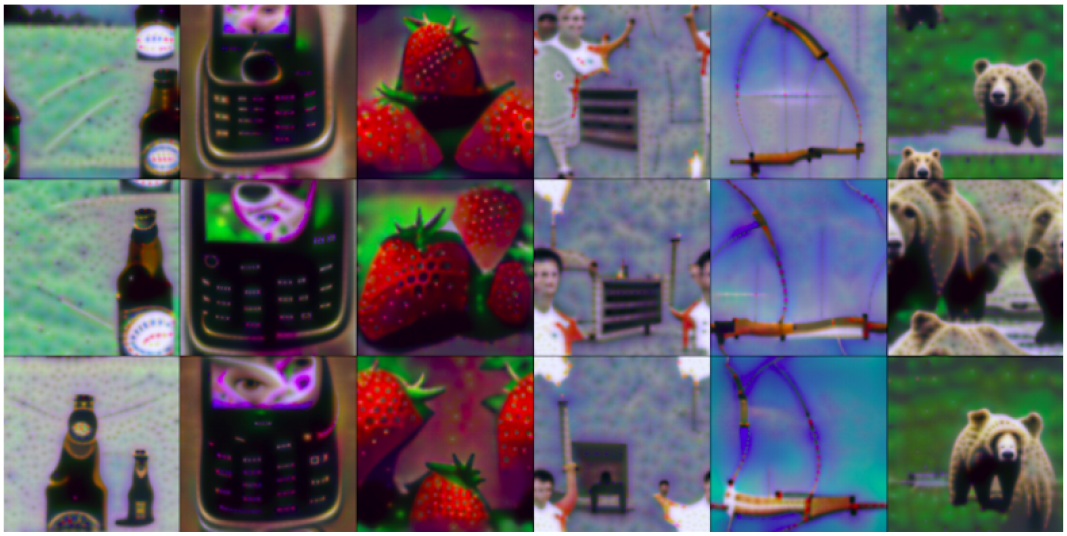
Zadnji eksperiment je pokretanje s dimenzijama 448x448 i β parametrima Adam optimizatora postavljenima na 0.9 i 0.999. S obzirom da su dimenzije velike, od svake klase se generiraju samo dva primjera. Na ovim slikama se vidi da je model generirao okruženje u kojima se generirane klase nalaze. Rezultati su prikazani na slikama 4.11 i 4.12. Boce stoje uspravno, jagode izgledaju kao da imaju zelene petiljke, baklje

Slika 4.7: Slike generirane s vrijednostima Adam optimizatora $\beta = 0,0$. Klase na slikama su redom po stupcima s lijeva na desno: pivska boca (engl. *beer bottle*), mobitel (engl. *cell*), jagoda (engl. *strawberry*), baklja (engl. *torch*), luk (engl. *bow*), smeđi medvjed (engl. *brown bear*)

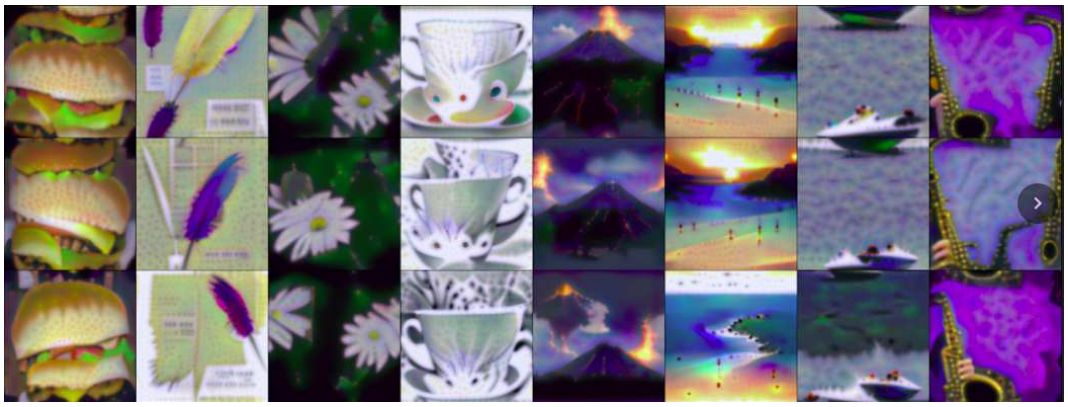
kao da su na nekom sportskom terenu, pera su na papirima i vulkani su u prirodnom okruženju.



Slika 4.8: Slike generirane s vrijednostima Adam optimizatora $\beta = 0,0$. Klase na slikama su redom po stupcima s lijeva na desno: cheeseburger, pero (engl. *quill*), tratinčica (engl. *daisy*), šalica (engl. *cup*), vulkan (engl. *vulcano*), plaža (engl. *seaside*), gliser (engl. *speedboat*) i saksofon (engl. *saxophone*)



Slika 4.9: Slike generirane s vrijednostima Adam optimizatora $\beta = 0,0$ i dimenzijama 336x336. Klase na slikama su redom po stupcima s lijeva na desno: pivska boca (engl. *beer bottle*), mobitel (engl. *cell*), jagoda (engl. *strawberry*), baklja (engl. *torch*), luk (engl. *bow*), smeđi medvjed (engl. *brown bear*)



Slika 4.10: Slike generirane s vrijednostima Adam optimizatora $\beta = 0,0$ i dimenzijama 336x336. Klase na slikama su redom po stupcima s lijeva na desno: cheeseburger, pero (engl. *quill*), tratinčica (engl. *daisy*), šalica (engl. *cup*), vulkan (engl. *vulcano*), plaža (engl. *seaside*), gliser (engl. *speedboat*) i saksofon (engl. *saxophone*)

Slika 4.11: Slike generirane s vrijednostima Adam optimizatora $\beta = 0.9, 0.999$ i dimenzijama 336x336. Klase na slikama su redom po stupcima s lijeva na desno: pivska boca (engl. *beer bottle*), mobitel (engl. *cell*), jagoda (engl. *strawberry*), baklja (engl. *torch*), luk (engl. *bow*), smeđi medvjed (engl. *brown bear*)

Slika 4.12: Slike generirane s vrijednostima Adam optimizatora $\beta = 0,0$ i dimenzijama 336x336. Klase na slikama su redom po stupcima s lijeva na desno: cheeseburger, pero (engl. *quill*), tratinčica (engl. *daisy*), šalica (engl. *cup*), vulkan (engl. *vulcano*), plaža (engl. *seaside*), gliser (engl. *speedboat*) i saksofon (engl. *saxophone*)

5. Implementacija

Sav kod napisan je u programskom jeziku Python 3. Korišten paket za duboko učenje je Pytorch Paszke et al. (2019), a za robusne modele korišten je paket robustness Engstrom et al. (2019).

Kao što je već navedeno, glavna ideja je minimizirati klasifikacijski gubitak uz dodane regularizacije. U kodu je samo potrebno definirati gubitke, a Pytorch, kad se kod pokrene, iz tih definicija računa gradijente tj. radi unazadnu propagaciju.

Za klasifikacijski gubitak, u svim eksperimentima je korišten gubitak unakrsne entropije, a u samom kodu korištena je klasa nn.CrossEntropyLoss iz paketa Pytorch. Potrebno je samo definirati očekivane klase i izlaz modela:

```
cross_entropy = nn.CrossEntropyLoss()
loss = cross_entropy(outputs, targets)
```

Gubitak *image prior* sastoji se od L2 regularizacije i regularizacije totalne varijacije. Za L2 regularizaciju korištena je funkcija *torch.norm* definirana u Pytorch paketu. Kod računanja norme grupe, izračunata je prosječna vrijednost L2 norme svake slike. Totalna varijacija, definirana (4.1) implementirana je funkcijom:

```
def get_total_variation(inputs):
    shift_0 = inputs[:, :, :-1, :] - inputs[:, :, 1:, :]
    shift_1 = inputs[:, :, :, :-1] - inputs[:, :, :, 1:]

    return torch.norm(shift_0) + torch.norm(shift_1)
```

Regularizacija pomoću normalizacije grupa ostvarena je uz pomoć Pytorch udica (engl. *hook*). Na svaki sloj normalizacije grupa stavljena je unaprijedna udica koja omogućava mjerenje aritmetičke sredine i varijance po kanalu sa svakim unaprijednim prolazom. Tijekom unaprijednog prolaza, svaka udica izračuna L2 norme aritmetičke sredine i varijance, a nakon prolaza se pročitaju vrijednosti sa svake udice i izračuna se suma svih L2 normi (3.7). Ova regularizacija je u kodu ostvarena pomoću klase za udicu prikazane u sljedećem odsječku koda i sumom vrijednosti regularizacija svih

udica.

```
class DeepInversionHook:
  def __init__(self, module, backward=False):
    if backward==False:
      self.hook = module.register_forward_hook(
        self.hook_fn)
    else:
      self.hook = module.register_backward_hook(
        self.hook_fn)

  def hook_fn(self, module, input, output):
    size = input[0].shape[1]
    mean = input[0].mean([0, 2, 3])
    var = input[0].permute(
      1, 0, 2, 3).contiguous().view(size, -1]).var(1)
    batch_mean = torch.norm(
      module.running_mean.data - mean, 2)
    batch_var = torch.norm(
      module.running_var.data - var, 2)

    self.regularization = batch_mean + batch_var

  def close(self):
    self.hook.remove()
```

Podrezivanje slika je implementirano kao što su Yin et al. (2020) opisali. U kodu je implementirano sa sljedećom funkcijom:

```
mean = np.array([0.485, 0.456, 0.406])
std = np.array([0.229, 0.224, 0.225])
def clip(inputs):
  for i, (m, s) in enumerate(zip(mean, std)):
    inputs[:, ] = torch.clamp(
      image_tensor[:, i], -m / s, (1 - m) / s)
    return image_tensor
```

Za rotaciju je korištena Pytorch funkcija roll:

```
inputs_rolled = torch.roll(
```

```
  inputs, shifts=(shift_0, shift_1), dims=(2, 3))
```

Kao optimizator je korišten Adam optimizator iz Pytorch biblioteke.

```
optimizer = optim.Adam(
  [inputs], lr=0.05, betas=[0.9, 0.999], eps = 1e-8)
```

6. Zaključak

U radu je prikazan postupak optimizacije klasifikacijskog gubitka i dodavanja različitih regularizacijskih izraza i njihovih efekata na generaciju slika te su reproducirane metode iz navedenih radova.

Na temelju dosadašnjih radova i potvrdom provedenih eksperimenata, može se zaključiti da se duboki modeli mogu vizualizirati te da se njihovo znanje može prikazati. Mana ovih metoda je što modeli ne razmišljaju na istoj razini kao i čovjek pa ono što se prikaže čovjeku ne može biti identično kao prava slika osim u slučaju kada je model previše prilagođen jednoj slici te je zapamtio tu sliku uz pomoć parametara. Uzevši to u obzir, metode daju dobru reprezentaciju slika koje, iako nisu skroz realne, su s lakoćom prepoznatljive ljudskom oku. Iz generiranih slika je jasno da model ima znanje o specifičnoj klasi kao cjelini, a ne da pamti samo skup jednostavnih značajki.

Iz eksperimenata se vidi da robusni modeli daju jasnije slike od nerobusnih. Pretpostavka je da su zbog otpornosti na perturbacije još manje ovisni o jednostavnim značajkama i da gledaju složenije značajke. Uz to daju i manje mutne slike.

Eksperiment sa slikama većih dimenzija daje drugačije rezultate. Jedini problem je što veće slike zahtijevaju puno više memorije pa su dimenzije u radu bile ograničene. Uz to se može primijetiti da se na tim slikama objekti pojavljuju više puta nego na slikama manjih dimenzija tj. da generirane slike neće biti skalirane verzije manjih slika, već prikazivati više jedinki odabrane klase.

Rad se kvantitativno može poboljšati eksperimentima s većim dimenzijama slika, eksperimentima s više iteracija i drugačijim vrijednostima hiperparametara te vizualizacijom nekih drugih arhitektura. Osim navedenog, bilo bi korisno usporediti sličnost generiranih slika sa slikama iz skupa za učenje.

LITERATURA

Iva Božić. Neprijateljski napadi na modele za klasifikaciju slike. stranice 18–20, 2019.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, i Dimitris Tsipras. Robustness (python library), 2019. URL `https://github.com/MadryLab/robustness`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 770–778, 2016.

Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL `http://arxiv.org/abs/1412.6980`. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, i Adrian Vladu. Towards deep learning models resistant to adversarial attacks. U *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`.

Alexander Mordvintsev, Christopher Olah, i Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, i Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. U H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, i R. Garnett, urednici, *Advances in Neural Information Processing Systems 32*, stranice 8024–8035. Curran

Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-lib pdf.

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, i Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, stranice 8715–8724, 2020.

Sažetak

Interpretiranje dubokih konvolucijskih modela je bitno za njihovo korištenje u industriji kako ne bi došlo do neočekivanih tj. neželjenih ishoda. U ovom radu se proučavaju i testiraju metode za interpretaciju modela. Metode obuhvaćene ovim radom vizualiziraju modele generiranjem slika korišenjem znanja modela. Pokazano je da napredne metode generiraju realistične slike i da korišteni modeli ne uče samo jednostavne značajke, već da mogu razaznati klase kao cjeline koje i čovjek vidi. U eksperimentima je pokazano da su robusni modeli još bolji u tome, odnosno da oni razaznaju značajke na višoj razini od nerobusnih modela te da povećanje dimenzija generiranih slika pomaže algoritmu za generiranje i da su samim time slike većih dimenzija realističnije od onih manjih dimenzija. Pokazani su rezultati eksperimenata i predloženi budući pravci razvoja.

Ključne riječi: računalni vid, interpretacija modela, inverzija modela, generacija slika, optimizacija ulaza

Interpreting Discriminative Convolutional Models

Abstract

Interpretation of deep convolutional models is essential for their use in industry in order to avoid unexpected, i.e. undesirable outcomes. This paper focuses on methods for model and tests them. The methods covered in this paper are used to visualize a model by generating images using the model knowledge. It has been shown that advanced methods generate realistic images and that the models used not only learn simple features, but that they can distinguish classes as wholes in a similar way a human can. Experiments have shown that robust models are even better at this, that they recognize features at a higher level than non-robust models, and that increasing the dimensions of generated images helps the generating algorithm and that thus larger images are more realistic than smaller ones. The results of the experiments are shown and future directions of development are suggested.

Keywords: computer vision, model interpretation, model inversion, image synthesis, input optimization