Interpretabilnost modela dubokog učenja

Ravlić, Domagoj

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Domagoj Ravlić

INTERPRETABILNOST MODELA DUBOKOG UČENJA

Diplomski rad

Voditelj rada:
dr.sc. Tomislav Šmuc

Zagreb, studeni, 2019.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Ovaj diplomski rad posvećujem svojoj obitelji koja je uvijek vjerovala u mene i koja mi je bila potpora na svakom koraku mog obrazovanja. Bez njih ovo ne bi bilo moguće. Zahvaljujem se mentorima dr.sc. Tomislavu Šmucu i dr.sc. Tomislavu Lipiću na svim uputama, savjetima i na svom znanju kojeg su mi prenijeli tijekom pisanja ovog diplomskog rada. Zahvaljujem se i svim prijateljima bez kojih bi ovaj period školovanja bio puno teži. Na kraju, zahvaljujem se svojoj Miji koja me je od prve godine pratila na ovom putovanju i uvijek bila uz mene kada god je to trebalo.

Sadržaj

Poglavlje 1

Uvod

Duboke neuronske mreže postižu zavidne performanse u područjima poput klasifikacije ili predikcije. U današnje vrijeme čak i dostižu sposobnosti samih ljudskih eksperata. Upravo zbog toga se sve više koriste u situacijama poput prepoznavanja objekta ili teksta na slici, generiranja *chat bot*-a, detektiranja *spam* poruka i slično. No, još uvijek se u najčešćem broju primjena ti modeli smatraju crnom kutijom za koje često ni sam autor ne može dovoljno dobro objasniti kako i zašto funkcionira. Samo je pitanje vremena kada će se duboke neuronske mreže početi koristiti u svrhe gdje je od presudne važnosti donijeti ispravnu odluku, ali i znati na koji način se donose te odluke, kao što je medicinska dijagnostika ili autonomna vožnja. Upravo zbog toga se javlja potreba za tehnikama koje će na ljudski razumljiv način interpretirati modele, odnosno omogućiti ljudsku/ekspertsku procjenu njihove robusnosti.

1.1 Pregled dosadašnjih istraživanja

Interpretabilnost modela dubokog učenja je od sve veće važnosti kako u teoriji, tako i u praksi. Sukladno tome, mnogi radovi su napisani na tu temu. Neki se bave interpretacijom već istreniranih modela, a neki treniranjem modela s interpretabilnim reprezentacijama. Sva ta istraživanja možemo ugrubo podijeliti u pet grupa:

- **vizualizacija CNN**[1] **reprezentacije** – najizravniji način pronalaska uzoraka koji doprinose rezultatu
- **dijagnoza CNN reprezentacije** – cilj je postići što bolji uvid u značajke koje su ”enkodirane” u konvolucijske neuronske mreže

[1] eng. Convolutional neural network.

- **"otpetljavanje" CNN reprezentacije u eksplanatorne grafove** – preciznije i robusnije metode od vizualizacije i dijagnoze konvolucijskih neuronskih mreža
- **učenje neuronske mreže koja ima interpretabilnu reprezentaciju** – metode orijentirane na treniranje konvolucijske neuronske mreže s reprezentacijom koja ima jasno semantičko značenje
- **interpretabilnost *middle-to-end* učenja** – metode koje se primjenjuju na modele učene *middle-to-end* metodom.

1.2 Cilj diplomskog rada

U ovom radu ćemo se baviti proučavanjem tehnika interpretabilnosti, uključujući vizualizaciju i dijagnostiku naučenih reprezentacija dubokih modela. Fokusirat ćemo se na konvolucijske neuronske mreže - CNN koje postižu impresivne rezultate na području računalnog vida. Cilj rada je primijeniti jednu od tih metoda nad postojećom konvolucijskom neuronskom mrežom i pokazati rezultate tog eksperimenta.

1.3 Struktura rada

Diplomski rad je podijeljen u pet poglavlja. U prvom poglavlju dajemo cilj diplomskog rada, motivaciju za interpretabilnost modela dubokog učenja te kratki pregled metoda koje se koriste u tu svrhu. U drugom poglavlju definiramo pojmove dubokog učenja i interpretabilnosti. U trećem poglavlju dajemo detaljan pregled najpoznatijih metoda interpretabilnosti konvolucijskih neuronskih mreža. U četvrtom poglavlju provodimo eksperimente u kojima koristimo dvije različite metode interpretabilnosti. Peto poglavlje je namijenjeno za zaključak.

Poglavlje 2

Duboko učenje

Duboko učenje je grana strojnog učenja koja je posebno prikladna za rješavanje problema iz područja umjetne inteligencije. Temelji se na predstavljanju podataka složenim reprezentacijama do kojih se dolazi slijedom naučenih nelinearnih transformacija što se postiže dubokim neuronskim mrežama.

2.1 Duboke neuronske mreže

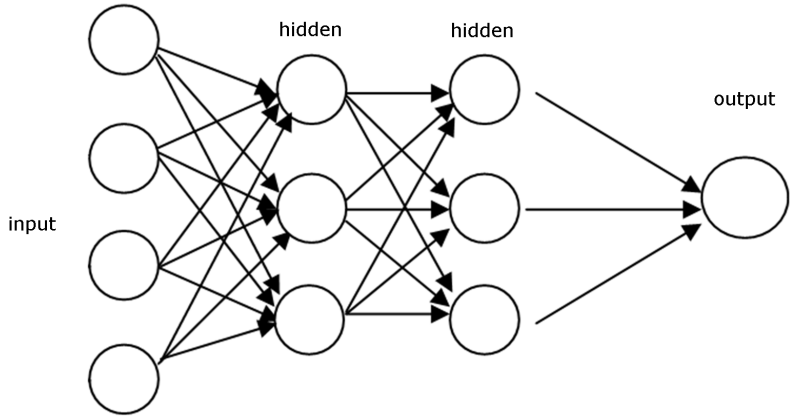
Osnovni primjer neuronske mreže je unaprijedna (eng. *feedforward*) neuronska mreža koju možemo reprezentirati pomoću acikličkog usmjerenog grafa. Čvorovi grafa raspoređeni su u ulazni, izlazni i u skrivene slojeve (eng. *hidden layers*). Ukoliko je prisutno dva ili više skrivenih čvorova, onda kažemo da se radi o *dubokoj neuronskoj mreži*. Svaki čvor određenog sloja je povezan sa svakim čvorom prethodnog sloja kako možemo vidjeti na slici 2.1 te kažemo su ti slojevi *potpuno povezani*.

Cilj neuronske mreže je pronaći aproksimaciju $h_\Theta \colon \mathbb{R}^n \to \mathbb{R}^m$ neke funkcije $f \colon \mathbb{R}^n \to \mathbb{R}^k$ koja preslikava ulaz $x \in \mathbb{R}^n$ u $y \in \mathbb{R}^k$ za neke $n, k \in \mathbb{N}$. Skup parametara mreže je definiran skupom $\Theta = \{\Theta^{(1)}, \Theta^{(2)}, \ldots, \Theta^{(L-1)}\}$ gdje $L \in \mathbb{N}$ predstavlja broj slojeva u neuronskoj mreži.

Za pronalazak aproksimacije, moramo imati dovoljno velik i reprezentativan skup ulaznih podataka X zajedno s njihovim stvarnim vrijednostima Y te *funkciju gubitka* koju ćemo označiti s $E \colon \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$. Ona mjeri razliku izmedu željene vrijednosti i naše aproksimacije s obzirom na trenutne parametre. Želimo minimizirati *funkciju cilja*, u oznaci $J(\Theta)$, koja sumira gubitke svih ulaznih vrijednosti iz X:

$$J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} E(h_\Theta(x^{(i)}), y^{(i)}) \tag{2.1}$$

gdje je $m \in \mathbb{N}$ veličina skupa X, odnosno Y.



Slika 2.1: Unaprijedna neuronska mreža na kojoj se jasno vidi raspodjela slojeva i povezanost čvorova.

Mreža ima slobodu iskoristiti skrivene slojeve na način koji osigurava najbolju aproksimaciju. Ona pomoću metode *gradijentnog spusta* iterativno pronalazi najbolje parametre $\theta \in \Theta^{(l)}$ za svaki $l = 1, 2, \ldots, l-1$ na način da ih ažurira s obzirom na negativni gradijent funkcije J:

$$\theta = \theta - \eta \frac{\partial}{\partial \theta} J(\Theta), \tag{2.2}$$

gdje je η stopa učenja (eng. *learning rate*) koja kontrolira promjenu parametra s obzirom na gradijent.

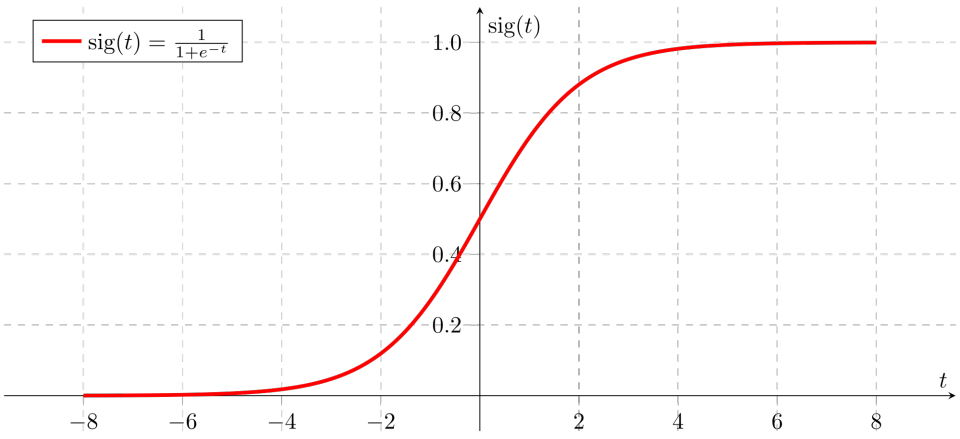
Nerijetko je potrebno aproksimirati nelinearne funkcije. Trenutno definirana mreža nije dobro rješenje za takvu aproksimaciju jer je aproksimacija h linearna funkcija. U tu svrhu se neuronima pridružuje aktivacijska funkcija na način da transformira vrijednost neurona i na taj način daje nelinearnost cijelom modelu. Najpoznatija takva aktivacijska funkcija je *sigmoid* funkcija

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.3}$$

čiji graf možemo vidjeti na slici 2.2.

2.2 Algoritam s povratnim postupkom

Zbog kompleksnosti modela opisanog u potpoglavlju Duboke neuronske mreže, potreban nam je efikasan način pronalaženja gradijenta koji će ažurirati parametre modela. Algori-



Slika 2.2: *Sigmoid* funkcija

tam s povratnim postupkom je u engleskoj literaturi poznat pod nazivom *backpropagation* i on upravo služi u takvoj situaciji.

Djeluje tako što se ulazni podatak propagira unaprijed kroz mrežu kako bi se dobio rezultat mreže. Nakon toga se određuje greška C_x koja je najčešće oblika $\frac{1}{2}\sum_k (a_k^{(L)} - y_k)^2$, gdje je $a_k^{(L)}$ aktivacija k-tog neurona zadnjeg sloja neuronske mreže L nakon aktivacijske funkcije, a y_k su stvarni rezultati na osnovu ulaznog podatka x. Jednom kada znamo veličinu greške, možemo povratnim postupkom prema prvom sloju odrediti koje težine $w_{j,k}^{(l)}$ i *bias*-e $b^{(l)}$ trebamo mijenjati kako bi se ta greška smanjila. Da napomenemo, ovdje $w_{j,k}^{(l)}$ označava težinu brida neuronske mreže koja spaja k-ti neuron $(l-1)$-og sloja s j-tim neuronom l-og sloja gdje je $l \in \{1, ..., L\}$, a $b^{(l)}$ označava *bias* u l-tom sloju. Cilj nam je pronaći

$$\nabla C_x = \begin{bmatrix} \frac{\partial C_x}{\partial b^{(1)}} \\ \frac{\partial C_x}{\partial w_{1,1}^{(1)}} \\ \vdots \\ \frac{\partial C_x}{\partial w_{j,k}^{(l)}} \\ \vdots \end{bmatrix} \tag{2.4}$$

za svaki ulazni podatak $x \in \mathbf{X}$. Kada pronađemo sve vektore ∇C_X, računamo njihovu srednju vrijednost u oznaci ∇C te ažuriramo sve težine i *bias*-e na osnovu tog vektora i stope učenja η. Sve što nam sada preostaje je izračunati sve $\frac{\partial C_x}{\partial b^{(l)}}$, $\frac{\partial C_x}{\partial w_{j,k}^{(l)}}$. Te vrijednosti računamo takozvanim *chain-link*-om.

Označimo s $z_k^{(l)} = \cdots + w_{j,k}^{(l)} * a_k^{(l-1)} + \cdots$ aktivaciju neurona prije aktivacijske funkcije. Računamo:

$$\frac{\partial C_x}{\partial w_{j,k}^{(l)}} = \frac{\partial z_j^{(l)}}{\partial w_{j,k}^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial C_x}{\partial a_j^{(l)}} \tag{2.5}$$

$$\frac{\partial C_x}{\partial b^{(l)}} = \frac{\partial z_j^{(l)}}{\partial b^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial C_x}{\partial a_j^{(l)}} \tag{2.6}$$

$$\frac{\partial C_x}{\partial a_k^{(l-1)}} = \sum_j \frac{\partial z_j^{(l)}}{\partial a_k^{(l-1)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial C_x}{\partial a_j^{(l)}}, \tag{2.7}$$

gdje su

$$\frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} = A'(z_j^{(l)}) \tag{2.8}$$

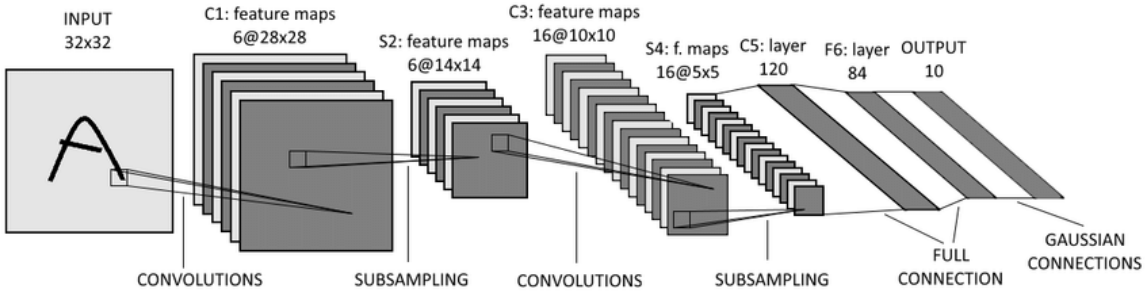
$$\frac{\partial z_j^{(l)}}{\partial w_{j,k}^{(l)}} = a_k^{(l-1)} \tag{2.9}$$

i gdje je A aktivacijska funkcija. $\frac{\partial C_x}{\partial a_k^{(l-1)}}$ je bitno izračunati kako bi mogli isti ovaj proces ponoviti za sljedeće (odnosno prethodne) slojeve.

2.3 Konvolucijske neuronske mreže

Konvolucijska neuronska mreža je nadogradnja unaprijedne neuronske mreže. Osim od ulaznog i izlaznog sloja te jednog ili više skrivenih potpuno povezanih slojeva, sastavljena je i od konvolucijskih slojeva te slojeva sažimanja koji obično dolaze na početak mreže (nakon ulaza). Najčešće prvo dolazi konvolucijski sloj pa nakon njega sloj sažimanja - i tako nekoliko puta. Nakon njih dolaze potpuno povezani slojevi.

Konvolucijske neuronske mreže su postale vrlo popularne u zadnjih par godina jer su se pokazale moćnim alatom u području računalnog vida. Arhitektura konvolucijskih neuronskih mreža pokazala se izrazito dobrom u radu sa slikama i prepoznavanju značajki s istih. Stoga se konvolucijska neuronska mreža nameće kao vrlo dobro rješenje pri prepoznavanju objekata na slici ili pri klasifikaciji slika. Na slici 2.3 je prikazana arhitektura jedne od najpoznatijih konvolucijskih neuronskih mreža *LeNet-5*.



Slika 2.3: Konvolucijska neuronska mreža "LeNet-5". Na slici se jasno vide konvolucijski slojevi i slojevi sažimanja koji dolaze na početku te nakon njih vidimo 3 potpuno povezana sloja. "LeNet-5" je poznata po svojim rezultatima nad MNIST skupom podataka rukom pisanih brojeva. Uspjela je postići točnost veću od 99%! Iako je osmišljena 1998. godine ([39]), popularnost je postigla pojavom moćnih procesora posljednih desetak godina.

2.3.1 Konvolucijski sloj

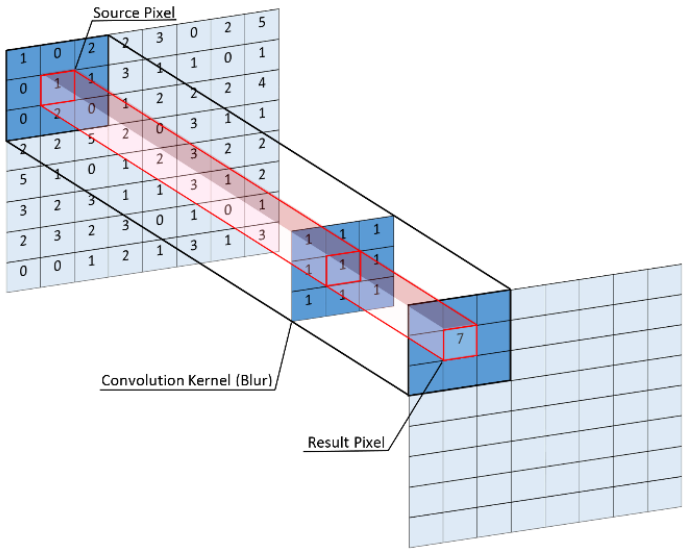
Konvolucijski sloj je temelj konvolucijske neuronske mreže. Sastoji se od skupa filtera kvadratnih dimenzija koje su manje od kvadratnih dimenzija ulazne slike. Na primjer, ukoliko je je ulazna slika dimenzije 25×25 modela RGB, tada konvolucijski sloj može sadržavati tri filtera dimenzija 5×5. Tri filtera su nužna za svaku komponentu boja ulazne slike. Filtere još zovemo *jezgra* i reprezentiramo ih preko kvadratnih matrica. Elemente te matrice konvolucijska neuronska mreža treba naučiti, odnosno pronaći, tijekom treniranja.

Proces konvolucije kreće od gornjeg lijevog kuta slike. Prije svega moramo znati pomak filtera (eng. *stride*) koji određuje za koliko piksela pomičemo filter s lijeva na desno krečući se kroz sliku. Nakon toga računamo prvi element mape značajki na način opisan slikom 2.4. Mapa značajki je ujedno i rezultat cijele konvolucije. Nakon što smo izračunali prvi element mape značajki, pomičemo filter udesno za onoliko piksela koliko smo definirali pomakom filtera te ponavljamo isti postupak. Kada dođemo do desnog dijela slike, pomičemo se za jedan piksel prema dolje i vraćamo filter na lijevi dio slike. Postupak prestaje kada dođemo do donjeg desnog dijela ulazne slike.

Dimenzija mape značajki O ovisi o pomaku filtera S, dimenziji ulazne slike I i o dimenziji filtera F. Računa se formulom:

$$O = \frac{I - F}{S} + 1. \tag{2.10}$$

Svrha konvolucijskog sloja je prepoznati značajke (poput rubova, kutova i slično) gdje god se oni nalazili na slici.



Slika 2.4: Primjer konvolucije. Pozicija filtera u procesu konvolucije prikazanog na slici je opisana engleskim terminom *Source Pixel*. Vrijednost svakog elementa u jezgri se množi vrijednošću piksela originalne slike s obzirom na *Source Pixel*. Zbroj tih umnožaka zapisujemo u mapu značajki.

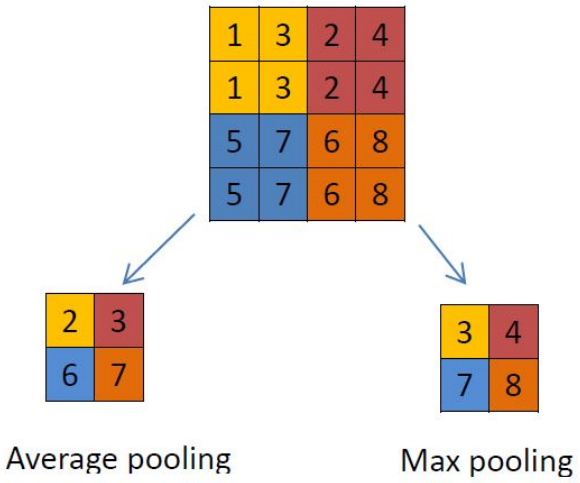
2.3.2 Sloj sažimanja

Slojevi sažimanja se koriste radi smanjenja rezolucije mapi svojstava u svrhu bržeg računanja te radi prostorne invarijantnosti – neosjetljivosti na manje pomake značajki na ulaznoj slici. Ovaj sloj također sadrži filter koji djeluje na drugačiji način nego u konvolucijskom sloju. Umjesto množenja, primjenjuje se funkcija koja za ulaz uzima određene vrijednosti piksela nekog dijela slike koji ima dimenzije kao i filter. Funkcija preslikava te vrijednosti piksela u neku drugu vrijednost. Najčešće se koriste sažimanja usrednjavanjem - *average pooling* i sažimanje maksimalnom vrijednošću - *max pooling*. Njihova djelovanja možemo vidjeti na slici 2.5. Kao i kod konvolucijskog sloja, potrebno je definirati pomak filtera.

2.3.3 Primjeri konvolucijskih neuronskih mreža

U ovom poglavlju ćemo ukratko opisati neke od konvolucijskih mreža koje ćemo koristiti u eksperimentalnom dijelu ovog rada. Radi se o jednima od najpopularnijih konvolucijskih neuronskih mreža, najčešće osmišljenih u svrhu natjecanja:

- **AlexNet** je jedna od prvih konvolucijskih neuronskih mreža koje su postale svjetski



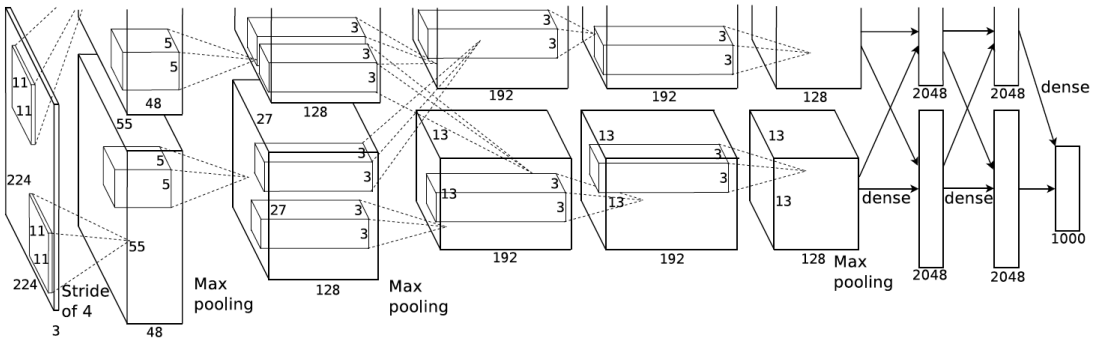
Slika 2.5: Average pooling i max pooling, jedne od najpopularnijih funkcija sažimanja, s pomakom vrijednosti dva. Obje funkcije u ovom slučaju kao ulaz primaju četiri prirodna broja i kao rezultat vraćaju srednje vrijednosti odnosno maksimalnu vrijednost ulaza.

popularne, kasnije ju je koristilo mnogo ljudi u svoje svrhe zbog njene efikasnosti. Dizajnirana je od strane Alexa Krizhevskya koji ju je onda i službeno objavio zajedno sa Sutskever I. i E. Hinton G. u radu [3]. Oni su osmislili *AlexNet* u svrhu *ImageNet LSVRC-2010* natjecanja održanog 2010. godine. Razni timovi su na tom natjecanju trenirali konvolucijsku neuronsku mrežu koja klasificira 1.2 milijuna slika u 1000 različitih kategorija. Na testnom skupu podataka, *AlexNet* je imala *top-1*[1] i *top-5*[2] grešku od 37.5%, odnosno 17.0%. Sastoji se od 60 milijuna parametara i 650000 neurona koji su razvrstani u 5 konvolucijskih slojeva, gdje nakon nekih dolaze slojevi sažimanja, te od 3 potpuno povezana sloja nakon kojeg slijedi *softmax* funkcija što možemo vidjeti na slici 2.6. Struktura *AlexNet* mreže koju danas poznajemo je zapravo varijacija spomenute konvolucijske neuronske mreže te je predstavljena na natjecanju *ImageNet LSVRC-2012*. To natjecanje je i osvojila s *top-5* greškom od 15.3%, što je velika razlika u usporedbi s drugoplasiranim CNN modelom koji je imao *top-5* grešku od 26.2%

- **VGG16** je konvolucijska neuronska mreža koja pripada porodici *VGGNet* konvolu-

[1]Greška nad nekim skupom podataka.

[2]Ukoliko se u 5 najvjerojatnijih klasa, dobivenih od strane CNN modela, nalazi i stvarna klasa tog ulaza onda se taj slučaj smatra uspješnim. *Top-5* greška pokazuje koliko je bilo takvih neuspješnih slučaja.

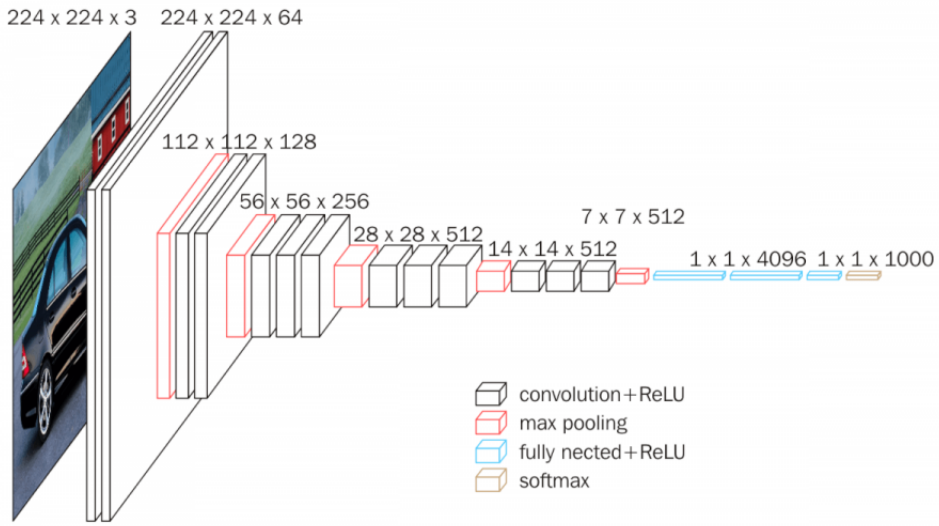


Slika 2.6: *AlexNet* je jedna od prvih popularnih konvolucijskih neuronskih mreža. Njena struktura se sastoji od 60 milijuna parametara i 650000 neurona koji su razvrstani u pet konvolucijskih slojeva, gdje nakon nekih dolaze slojevi sažimanja, te od tri potpuno povezana sloja nakon kojeg slijedi *softmax* funkcija. Izvor: [3].

cijskih neuronskih mreža, a one su poznate po tome što su dublje od *AlexNet* mreže – nakon dva konvolucijska sloja dolazi *max pooling* sloj sažimanja te su svi filteri dimenzije 3×3. *VGG16* je predstavljena 2014. godine na *ImageNet LSVRC*-2014 natjecanju te je osvojila drugo mjesto u području klasifikacije s *top-5* greškom od 7.3%. Sama mreža ima 144 milijuna parametara te se sastoji od 16 konvolucijskih slojeva i pet slojeva sažimanja s filterom dimenzije 2×2. Na kraju se nalaze tri potpuno povezana sloja praćena *softmax* funkcijom. Na sve skrivene slojeve se primjenjuje *ReLu* aktivacijska funkcija. Cijelu strukturu mreže možemo vidjeti na slici 2.7.

- **GoogleNet** ([8]), poznata i pod nazivom *Inception*, je konvolucijska neuronska mreža nastala na natjecanju *ILSVRC* 2014 na kojem je ujedno i odnijela pobjedu s *top-5* greškom od 6.7%. Arhitektura mreže je inspirirana *LeNet-5* konvolucijskom neuronskom mrežom, a sadrži oko pet milijuna parametara što je 12 puta manje nego što ima *AlexNet* konvolucijska neuronska mreža. Također, sadrži samo 22 sloja te se trenira brzo u odnosu na ostale dotadašnje mreže.

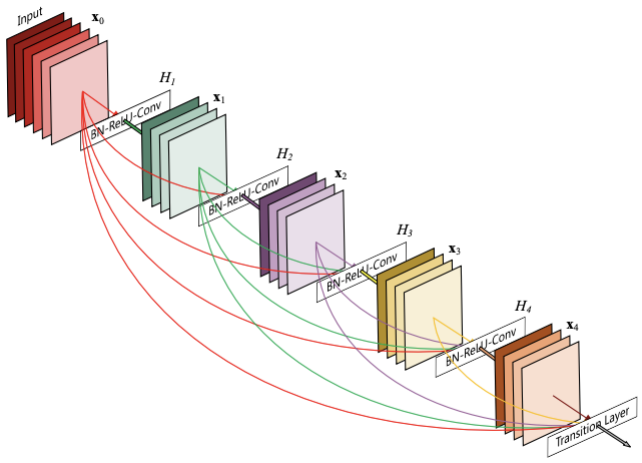
- **ResNet101** je razvijena 2015. godine u svrhu natjecanja *ILSVRC 2015* na kojem je i pobijedila u području klasifikacije s *top-5* greškom od 3.57%. Rezidualne mreže ([19]) su se pojavile kao rješenje problema treniranja vrlo dubokih konvolucijskih neuronskih mreža. Naime, do tada se točnost u nekim slučajevima nije povećala s većim brojem slojeva kako se očekivalo. Rezidualne mreže su riješile taj problem op-



Slika 2.7: *VGG16* konvolucijska neuronska mreža pripada porodici *VGGNet* ([21]) konvolucijskih neuronskih mreža koje su poznate po tome što su dublje od *AlexNet* mreže, nakon dva konvolucijska sloja dolazi *max pooling* sloj sažimanja te su svi filteri dimenzije 3×3. Sama mreža ima 144 milijuna parametara te se sastoji od 16 konvolucijskih slojeva, pet slojeva sažimanja s filterom dimenzije 2×2. Na kraju se nalaze 3 potpuno povezana sloja praćeni *softmax* funkcijom. Na sve skrivene slojeve se primjenjuje *ReLu* aktivacijska funkcija. Izvor: [37].

timizacije uvođenjem takozvanih rezidualnih blokova koji mogu sadržavati "prečac povezanost" koja preskače pojedine slojeve. Struktura mreže se sastoji od 152 sloja što je otprilike osam puta više slojeva nego što to ima CNN model iz *VGGNet* porodice, ali treba naglasiti kako su He *et al.* uspjeli optimizirati trening ove mreže koji traje manje od treninga neke *VGGNet*.

- **DenseNet161** je nastala 2017. godine ([13]) kao poboljšanje *ResNet* konvolucijskih neuronskih mreža. Huang *et al.* su u svome radu pokazali da mreže mogu biti znatno dublje, točnije i učinkovitije za treniranje ako sadrže kraće veze između slojeva blizu ulaza i slojeva blizu izlaza. Sloj u *DenseNet161* je povezanih sa svakim slojem koji dolazi nakon njega u mreži što možemo vidjeti na slici 2.8. Na taj način mreža s L slojeva ima $\frac{L(L+1)}{2}$ povezanih slojeva umjesto L u prethodnim konvolucijskim neuronskim mrežama. Zbog takve arhitekture, automatski imamo manje parametara nego u tradicionalnim CNN modelima jer nema potrebe za učenjem redundantnih mapi značajki.



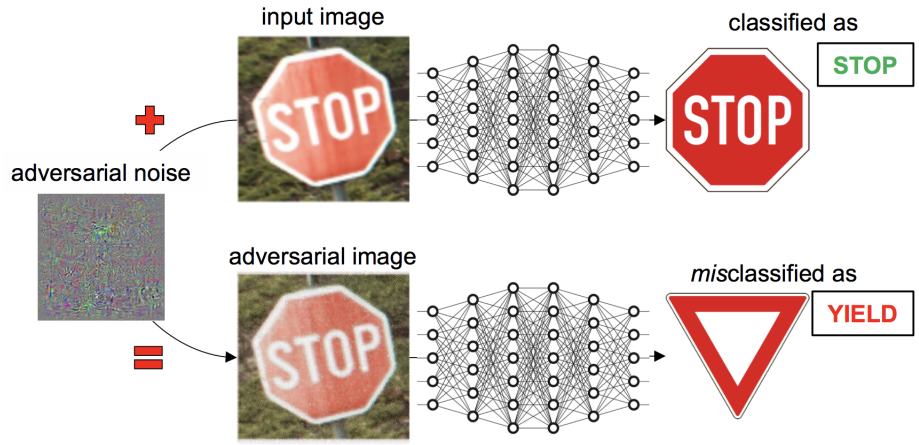
Slika 2.8: Slika koja ilustrira način na koji se povezuju slojevi u *DenseNet161* konvolucijskoj neuronskoj mreži.

- **SqueezeNet** ([12]) je mreža koja postiže sličnu točnost kao i *AlexNet* konvolucijska neuronska mreža, ali je dosta manja od nje. Osmišljena je 2016. godine te joj je cilj bio postići relativno visoku točnost na *ImageNet* skupu podataka s vrlo niskim brojem parametara u svrhu lakšeg treniranja konvolucijske neuronske mreže u distribuiranim sustavima, bržeg prenošenja informacija u autonomnoj vožnji te sličnih stvari. Uspjeli su u tom cilju jer su postigli rezultate slične kao i *AlexNet* CNN model. Zanimljivo je da *SqueezeNet* ima 50 puta manji broj parametara nego *AlexNet*, a pomoću kompresijskih algoritama se njezina veličina može spustiti na manje od 0.5MB što je za otprilike 150 puta manja veličina nego kod *AlexNet* konvolucijske neuronske mreže.

2.4 Interpretabilnost

Modeli bazirani na dubokim neuronskim mrežama postižu izvanredne rezultate u raznim područjima, ali metrike koje opisuju njihove rezultate, poput točnosti, osjetljivosti ili preciznosti, nekad nisu dovoljno pouzdani pokazatelji za stvarni svijet gdje je nekad od iznimne važnosti donijeti ispravnu odluku. Sasvim je razumno osmisliti model koji prepoznaje rukom pisane brojeve, poput *LeNet-5* modela, i primijenjivati ga u praksi samo na osnovu njegove točnosti nad velikim skupom podataka. Njegova greška neće toliko utjecati na korisnika tog modela. No, kada se modeli strojnog učenja primjenjuju u auto-

mobilskoj ili vojnoj industriji, medicini ili pri zaštiti podataka, onda je korisnicima takvih modela važno znati puno više od standardnih metrika o samoj funckionalnosti modela. Na slikama 2.9 i 2.10 možemo vidjeti na kakve probleme može naići mreža koja ima visoku točnost nad velikim brojem primjera. Krive klasifikacije, poput tih na slikama, u autonomnoj vožnji mogu dovesti čak i do smrti putnika u vozilu. Samo na ovim primjerima vidimo koliko je važno znati na koji način model strojnog učenja prepoznaje odnosno klasificira objekte.



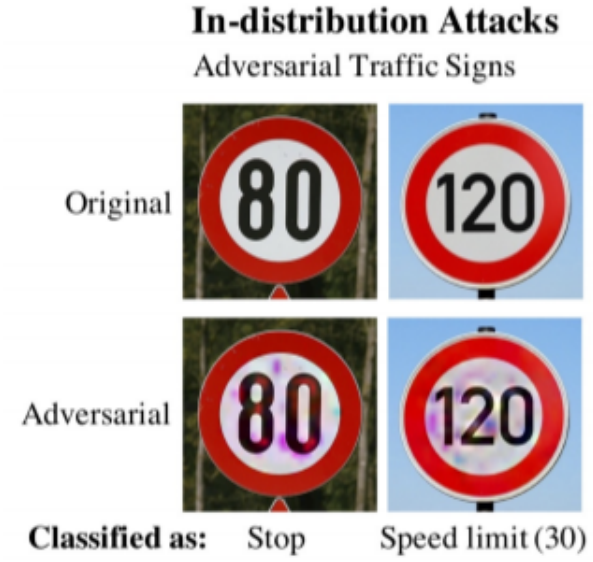
Slika 2.9: Pogrešno klasificiran znak STOP. Određeni šum može ”navesti” mrežu na krivu klasifikaciju. U ovom primjeru ista konvolucijska neuronska mreža dva naizgled jednaka znaka ne prepoznaje kao iste.

Možemo reći da je interpretabilnost modela itekako korisna u sljedeća tri slučaja vezana uz umjetnu inteligenciju.

Prvi slučaj obuhvaća modele koji su lošiji od čovjeka po pitanju performansi i koji još uvijek nemaju odlične rezultate koji bi ljudima dali povod da ih intenzivno primjenjuju (npr. *visual question answering* opisan u [1]). Ovdje interpretabilnost može pomoći znanstvenicima, koji razvijaju takav model, u lakšem pronalaženju grešaka čime se postiže veće razumijevanje problema te brži napredak u razvoju modela.

Drugi slučaj obuhvaća modele koji imaju otprilike iste performanse kao i prosječna osoba te se već neko vrijeme primjenjuju u praksi (npr. model koji klasificira slike na nekom konačnom broju kategorija, učen nad dovoljnom količinom podataka). Interpretabilnost je važna i u ovom slučaju jer utječe na povjerenje samog korisnika tog modela.

U treći slučaj spadaju modeli koji su premašili performanse samog čovjeka (npr. šah ili Go čiji model je opisan u [10]). Cilj interpretabilnosti je naučiti čovjeka neke nove stvari koje model obavlja.



Slika 2.10: Pogrešno klasificirani znakovi za ograničenje brzine na cesti. Slikama u drugom redu je dodan šum te vidimo kako nakon njega mreža krivo klasificira takve slike. Znak za ograničenje brzine od 80 km/h prepoznaje kao znak STOP, a znak za ograničenje brzine od 120 km/h prepoznaje kao ograničenje brzine od 30 km/h.

Također, društvo će prije prihvatiti umjetnu inteligenciju ako ona sa sobom donosi objašnjenje svoga rada. Upravo zbog toga je važno istraživati/unaprijeđivati metode interpretabilnosti.

Poglavlje 3

Metode interpretabilnosti konvolucijskih neuronskih mreža

U ovom poglavlju ćemo napraviti detaljniji pregled metoda interpretabilnosti CNN-a. Poglavlje je podijeljeno u pet potpoglavlja opisanih u Pregled dosadašnjih istraživanja.

3.1 Vizualizacija CNN reprezentacije

Vizualizacija filtera u konvolucijskim neuronskim mrežama je najizravniji način pronalaska uzoraka koji doprinose rezultatu. Različite metode su razvijene u ovu svrhu.

Najpoznatije su **gradijentne** metode ([25], [4], [18]) te metoda dana u radu Simonyan *et al*. ([20]) koju ćemo detaljnije proučiti u ovom poglavlju. Osim gradijentnih metoda, vrlo je poznata metoda naziva ***up-convolutional net*** obrađena u [2]. Ta metoda može biti dobar alat za vizualizaciju jer invertira mape značajki u slike. Problem ove metode je što matematički ne može osigurati da njezina vizualizacija stvarno odgovara CNN reprezentaciji, za razliku od gradijentne metode.

Simonyan je predstavio metodu koja može pronaći sliku koja maksimizira rezultat određene klase, odnosno predstavlja tu klasu. Preciznije, neka je $S_c(I)$ rezultat klase c dobiven istreniranom konvolucijskom neuronskom mrežom s ulaznom slikom I. Cilj nam je pronaći sliku I koja će maksimizirati rezultat $S_c(I)$ na osnovu sljedećeg izraza:

$$\underset{I}{\operatorname{argmax}}\, S_c(I) - \lambda \|I\|_2^2 , \tag{3.1}$$

gdje je λ regularizacijski parametar. Sliku I lokalno možemo pronaći algoritmom s povratnim postupkom. Algoritam funckionira istim principom kao i kod treniranja neuronske mreže, samo što su, u ovom slučaju, težine u mreži fiksne, a ono što se algoritmom pronalazi su vrijednosti piksela slike I. Na početku algoritma, slika I je inicijalizirana sa

srednjom vrijednosti piksela svih slika trening skupa. Neki primjeri slika koje predstavljaju određene klase možemo vidjeti na slici 3.1.

Treba napomenuti kako S_c rezultati nisu regularizirani s funkcijom koja izlazne rezultate svih klasa preslikava u vrijednosti koje u zbroju daju jedan. Primjer jedne takve funkcije je *softmax* funkcija $P_c = \frac{\exp S_c}{\sum_c \exp S_c}$. Razlog tome je što u suprotnom slučaju algoritam s povratnim postupkom teži minimiziranju rezultata ostalih klasa umjesto da se usredotoči na promatranu klasu. Također, Simonyan *et al.* su provodili eksperimente i sa *softmax* funkcijom prisutnom u zadnjem sloju, ali ti eksperimenti nisu davali dobre rezultate.

Simonyan *et al.* su također proučavali takozvane *saliency* mape. Te mape govore koji pikseli slike I_0 najviše utječu na promjenu rezultata $S_c(I_0)$ neke klase c. Prije nego opišemo postupak pronalaska te mape, promotrimo sljedeći linearni model kao motivacijski primjer:

$$S_c(I) = w_c^T * I + b_c, \tag{3.2}$$

gdje je slika I jednodimenzionalan vektor, w_c vektor težina i b_c vektor pomaka (eng. *bias*). U ovom slučaju je lagano primijetiti kako vektor w_c definira utjecaj piksela slike I na rezultat $S_c(I)$. No, u slučaju konvolucijskih neuronskih mreža, jasno je da je $S_c(I)$ kompleksna nelinearna funkcija te prethodno razmišljanje ne možemo samo tako primijeniti na ovaj slučaj. Ali, za danu sliku I_0, možemo aproksimirati rezultat $S_c(I_0)$ linearnom funkcijom u okolini od I_0 razvijanjem Taylorovog polinoma prvog stupnja oko te točke (slike I_0):

$$S_c(I) \approx w^T * I + b, \tag{3.3}$$

gdje je w derivacije funkcije S_c u smjeru slike I u točki (slici) I_0:

$$w = \left.\frac{\partial S_c}{\partial I}\right|_{I_0}. \tag{3.4}$$

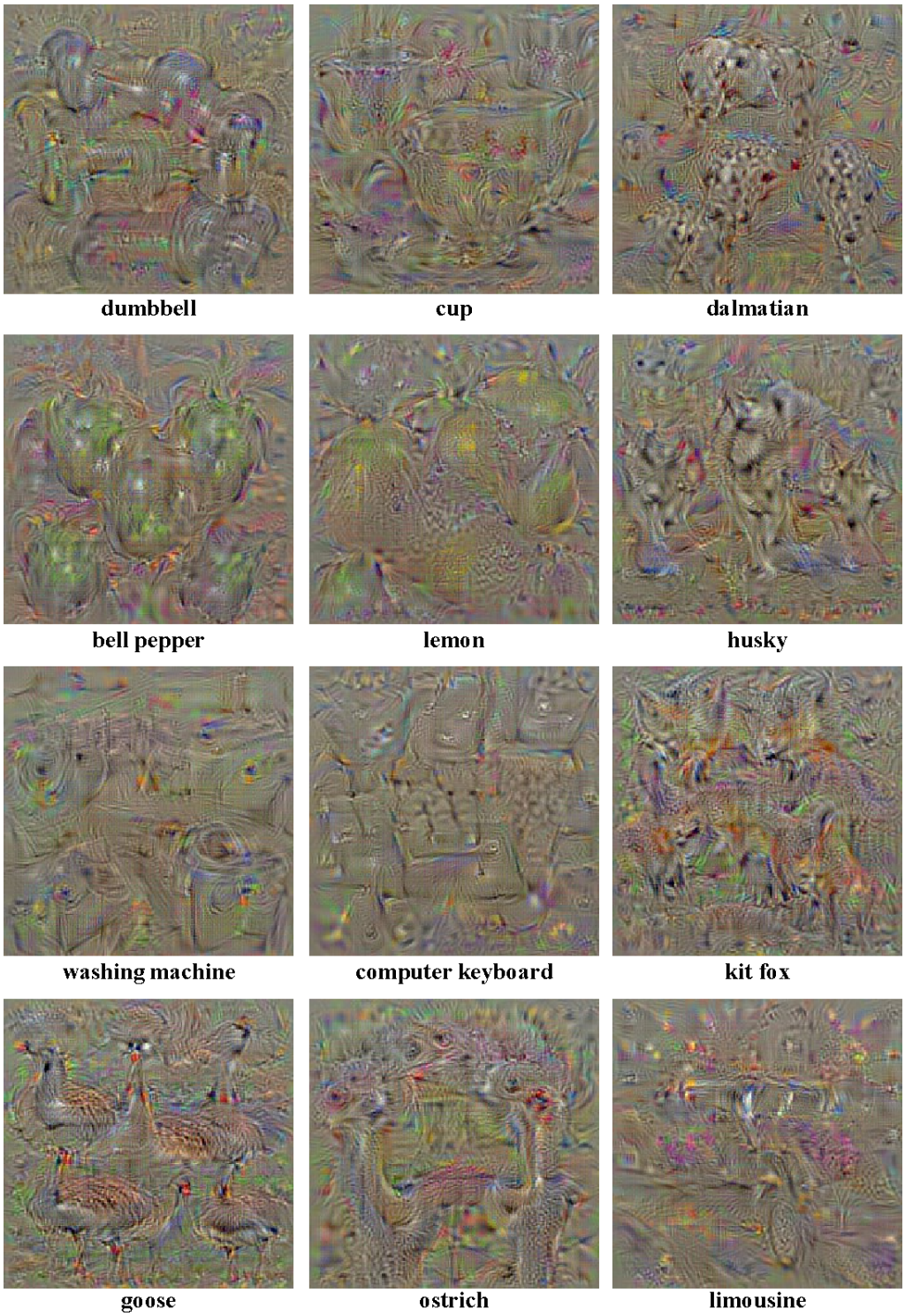
Postupak pronalaženja *saliency* mape $M \in \mathbb{R}^{m \times n}$ s m redova i n stupaca na osnovu ulazne slike I_0 za neku klasu c je sljedeći. Prvo moramo pronaći parcijalnu derivaciju w (3.4) pomoću algoritma s povratnim postupkom. Nakon toga su moguća dva slučaja. Ukoliko se radi o slici koja ima samo sivu komponentu boja, broj elemenata vektora w je jednak broju piksela slike I_0 pa se mapa M može izračunati kao:

$$M_{i,j} = |w_{h(i,j)}|, \tag{3.5}$$

gdje je $h(i, j)$ indeks elementa vektora w koji odgovara pikselu slike I_0 u i-tom retku i j-tom stupcu.

Ukoliko se radi o *RGB* slici koja ima tri komponente boja, postpupak je malo drugačiji. Neka piksel komponente boje c na poziciji (i, j) odgovara elementu vektora w na poziciji $h(i, j, c)$. Tada mapu M računamo kao:

$$M_{i,j} = \max_c |w_{h(i,j,c)}|. \tag{3.6}$$



Slika 3.1: Slike koje reprezentiraju pojedine klase. Na slikama možemo vidjeti očite oblike koje predstavljaju pojedinu klasu. Primijetimo kako su različiti aspekti klase sadržani u jednoj slici. Izvor: [20]

Potrebno je napomenuti kako za ovaj proces nije potrebno dodatno označavati podatke te da je računanje vrlo brzo pošto je potrebno samo jednom pokrenuti algoritam s povratnim postupkom! Primjere *saliency* mapa možemo vidjeti na slici 3.2.

3.2 Dijagnoza CNN reprezentacije

Cilj dijagnoze CNN reprezentacije je postići što bolji uvid u značajke "enkodirane" u konvolucijske neuronske mreže.

Szegedy *et al.* [9] su istraživali semantička značenja svakog filtera konvolucijske neuronske mreže. Aubry i Russell [24] te Lu [40] su pronalazili distribucije značajki različitih kategorija/atributa u prostoru značajki već istranirane konvolucijske neuronske mreže.

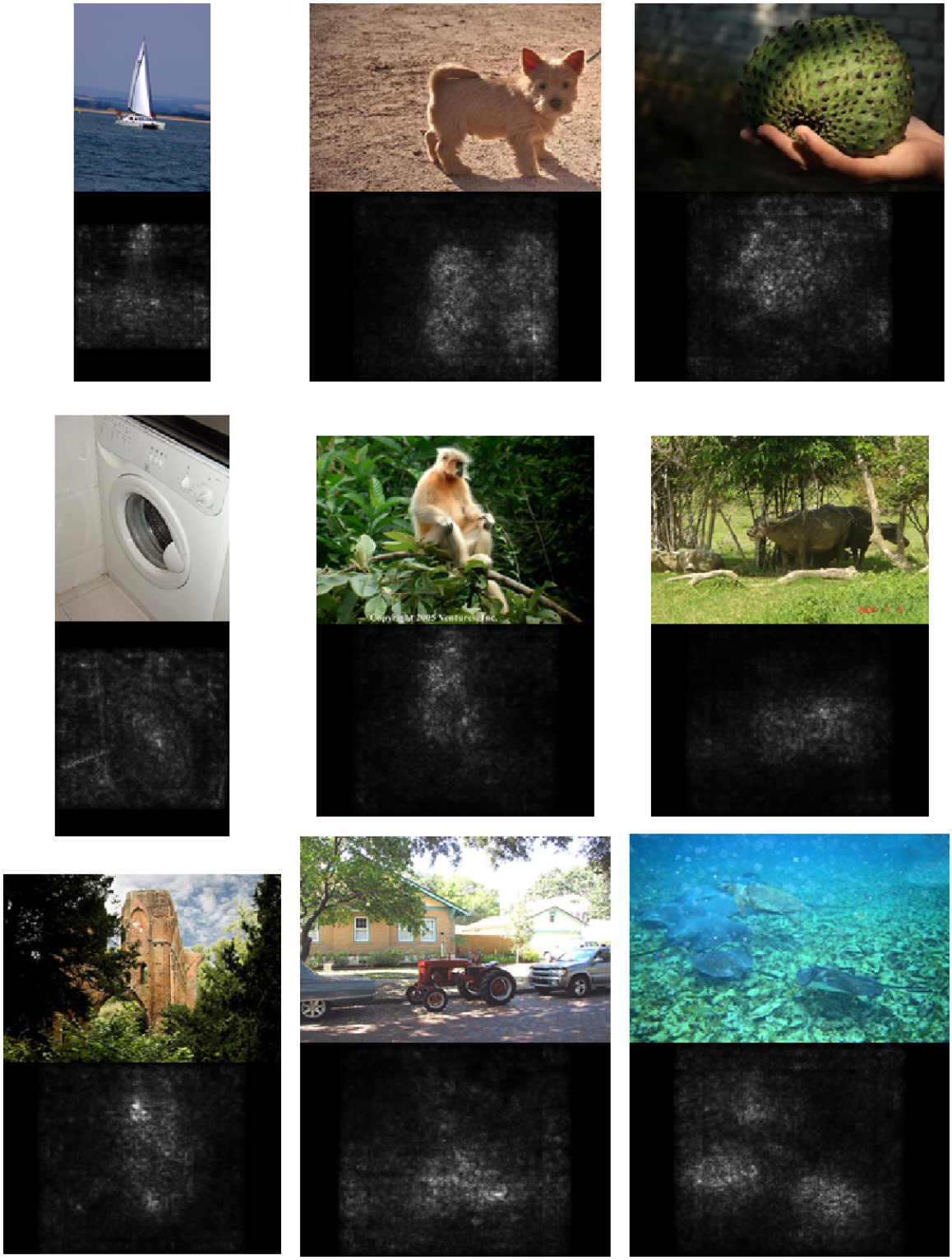
Pronalazak "ranjivih točki" u prostoru značajki CNN modela je čest pristup pri postavljanju dijagnoze. Su *et al.* [16], Koh i Liang [27] te Szegedy *et al.* [9] su predložili metode koje pronalaze minimalan šum na ulaznoj slici koji mijenja rezultat predikcije. Koh i Liang su također predstavili utjecajne funkcije koje omogućuju kreiranje novih slika koje sadrže takav određeni šum te pomoću njih možemo nadograđivati trening skup, ali i uspješnije *debuggirati* CNN model.

Prostor značajki su također proučavali i Lakkaraju *et al.* [14] koji su predložili metodu za otkrivanje nepoznatih uzoraka CNN modela. Ta metoda grupira sve točke prostora značajki u tisuće pseudo-kategorija i pretpostavlja da će dobro istrenirana konvolucijska neuronska mreža koristiti potprostor svake pseudo-kategorije kako bi ispravno reprezentirala podskup objekata neke klase. Na taj način se lako mogu otkriti mane CNN modela.

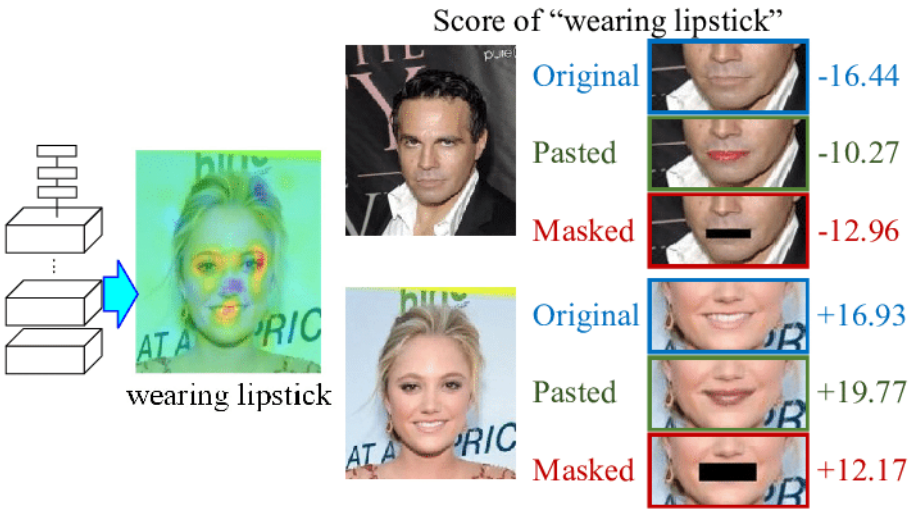
Zhang *et al.* [32] su osmislili metodu koja otkriva potencijalne sklonosti (*bias*) CNN modela nad trening skupom podataka. Na slici 3.3 možemo vidjeti primjer takve situacije.

Osmišljene su i metode ([34], [23], [28]) koje računaju gradijente mapi značajki s obzirom na grešku CNN modela kako bi procijenili regije na slici koje najviše doprinose rezultatu konvolucijske neuronske mreže. Jedne od takvih najpopularnijih metoda su *LIME* metoda ([26]) i ***Grad-CAM*** ([35]) koju ćemo detaljnije opisati u ovom poglavlju.

Google Brain tim na čelu sa znanstvenicom Been Kim je 2017. godine osmislio metodu naziva *TCAV* ([6]) koja koristi takozvane *CAV* vektore koji služe boljem razumijevanju mreže u smislu koncepata koje ljudi razumiju. Prednost ove metode je što ju mogu koristiti i osobe koje nemaju ili imaju minimalno znanje o strojnom učenju općenito. Također, metodi je moguće zadati bilo koji koncept (npr. spol, rasu ili slično) te ne zahtijeva dodatno treniranje niti mijenjanje promatranog CNN modela. Ovu metodu ćemo također detaljnije opisati u ovom poglavlju.



Slika 3.2: *Saliency* mape. Primijetimo kako su najvažniji pikseli oni koji karakteriziraju objekt koji prepoznajemo poput guma traktora ili vrata perilice rublja. Izvor: [20].



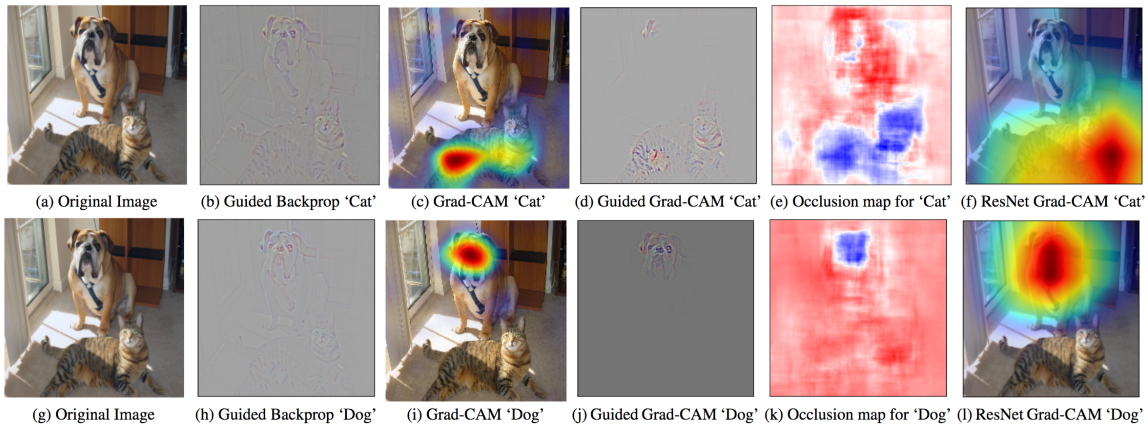
Slika 3.3: Primjer sklonosti. Promatramo originalnu sliku, sliku gdje je ruž umjetno stavljen na usta i sliku gdje su usne prekrivene crnim pravokutnikom. Naime, model je treniranjem shvatio kako se ruž za usne najčešće pojavljuje uz ostalu šminku na licu. Upravo zbog toga će za mušku osobu, koja nema šminku na sebi, u sva tri slučaja pretpostaviti da osoba ne nosi ruž za usne (negativan *score*). Za žensku osobu koja nosi maskaru će pretpostaviti suprotno u sva tri slučaja (pozitivan *score*). Izvor: [32].

3.2.1 Grad-CAM

Selvaraju *et al.* su najviše bili potaknuti *CAM* metodom ([7]) pri osmišljavanju *Grad-CAM* metode. Naime, *CAM* metoda također procjenjuje regije na slici koje najviše doprinose rezultatu konvolucijske neuronske mreže (lokalizacija). Mana te metode je što očekuje specifičnu arhitekturu CNN modela, odnosno preduvjet za provođenje metode je da prije *softmax* sloja dolazi konvolucijski sloj pa nakon njega sloj sažimanja. Naravno, svaka konvolucijska neuronska mreža se može transformirati u taj oblik mijenjanjem potpuno povezanih slojeva u konvolucijske slojeve i slojeve sažimanja. Taj preduvjet je ujedno i velika mana ove metode jer takve arhitekture CNN modela su nerijetko inferiornije u smislu točnosti naspram arhitektura koje prije *softmax* sloja imaju potpuno povezani sloj. Također, mana metode je što se mreža mora modificirati ako ne zadovoljava te uvjete.

U konačnici, Selvaraju *et al.* su osmislili metodu koja je generalizacija *CAM* metode koja funkcionira sa svim arhitekturama konvolucijskih neuronskih mreža. Oni tvrde da su razni prijašnji radovi zaključili kako se duboka reprezentacija CNN modela najbolje prikazuje kroz konvolucijske slojeve koji prikazuju značajke viših razina. Nadalje, poznato je da se informacija o tim značajkama gubi u potpuno povezanim slojevima pa su se fokusi-

rali na zadnji konvolucijski sloj koji sadrži najviše informacija o značajkama više razine, ali i o samoj semantici djelovanja CNN modela. Iako se gradijenti mapi značajki pomoću *Grad-CAM*-a mogu računati u svakom sloju, upravo zbog navedenih razloga se računaju u zadnjem sloju jer je cilj *Grad-CAM* što bolje interpretirati CNN model.



Slika 3.4: Grad-CAM. Slike a) i g) su originalne slike mačke i psa. Prvi red se odnosi na klasifikaciju mačke, a drugi na klasifikaciju psa. Slike b) i h) su dobivene *Guided backpropagation* metodom ([18]). Slike c) i i) predstavljaju rezultat *Grad-CAM* metode. Slike d) i j) su dobivene produktom slika dobivenih metodama *Guided backpropagation* i *Grad-CAM* te se takva metoda naziva *Guided Grad-CAM*. Slike e) i k) pokazuju dokaze prisutnosti neke klase označene plavom bojom. Na kraju, slike f) i l) prikazuju rezultate *Grad-CAM* metode na popularnoj ResNet konvolucijskoj neuronskoj mreži. Izvor: [35].

Takozvanu lokalizacijsku mapu $L^c_{\text{Grad-CAM}} \in \mathbb{R}^{u \times v}$ s brojem redaka u i brojem stupaca v, koja je ujedno i rezultat *Grad-CAM* metode, računamo na sljedeći način.

Prvo računamo gradijent rezultata y^c (prije *softmax* funkcije) neke klase c u smjeru mape značajki A^k zadnjeg konvolucijskog sloja (*i.e.* $\frac{\partial y^c}{\partial A^k}$). Tada računamo utjecaj α^c_k mape značajki k neke klase c s globalnim sažimanjem

$$\alpha^c_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A^k_{i,j}}, \tag{3.7}$$

gdje se gradijenti $\frac{\partial y^c}{\partial A^k_{i,j}}$ dobiju pomoću algoritma s povratnim postupkom.

Nakon toga možemo izračunati $L^c_{\text{Grad-CAM}}$ pomoću formule

$$L^c_{\text{Grad-CAM}} = ReLU\Big(\sum_k \alpha^c_k A^k\Big). \tag{3.8}$$

Primijetimo kako je lokalizacijska mapa $L^c_{\text{Grad-CAM}}$ dimenzije iste kao i mape značajki A^k. Također, *ReLU* se primjenjuje na kraju jer se želimo fokusirati samo na značajke koje pozitivno utječu na rezultat konvolucijske neuronske mreže, odnosno piksele čiji se intenzitet treba povećati kako bi se krajnji rezultat y^c povećao. Pikseli čiji se intenzitet treba smanjiti u slučaju povećavanja krajnjeg rezultata su pikseli koji najvjerojatnije pridonose rezultatu neke druge klase te nas oni u tom slučaju ne zanimaju. Bez *ReLU* funkcije, rezultati i lokalizacija nisu bili dobri. Slike c), f), i) i l) na 3.4 prikazuju vizualizaciju Grad-CAM metode.

Iako *Grad-CAM* ima mogućnost razlikovanja klasa i lokalizacije relevantnih regija slike pripadne klase, nedostaje mu preciznost u sitnim detaljima. Primjer možemo vidjeti na slici c) na slici 3.4. Naime, jasno se vidi kako *heat* mapa odgovara tigrastoj mački, ali je nejasno zašto se baš radi o tigrastoj mački. Drugim riječima, nedostaje nam vidljiv prikaz sitnih detalja (poput pruga na mački) kako bi bilo prilično jasno zašto se određena klasifikacija dogodila. Selvaraju *et al*. su se zbog toga okrenuli metodama poput *Guided backpropagation* ([18]) i *Deconvolution* ([25]). Naime, te metode su slične gradijentnim metodama spomenutima u potpoglavlju Vizualizacija CNN reprezentacije osim što one modificiraju gradijent što rezultira kvalitetnijim rezultatom u smislu sitnih detalja. Mana tih metoda je što ne mogu razlikovati klase, što se može vidjeti na slikama b) i h) na slici 3.4. Selvaraju *et al*. su odlučili iskoristiti prednosti metode *Guided backpropagation* te su osmislili metodu ***Guided Grad-CAM*** koja se dobije produktom rezultata metode *Guided backpropagation* i metode *Grad-CAM*. Ukoliko rezultat odnosno lokalizacijska mapa metode *Grad-CAM* ima manje dimenzije nego rezultat metode *Guided backpropagation*, koji je visoke rezolucije, onda se lokalizacijska mapa *up-sample*-a do željene dimenzije pomoću bilinearne interpolacije. Primjer djelovanja metode *Guided Grad-CAM* se može vidjeti na slikama d) i j) na slici 3.4. Rezultat je visoke rezolucije (mačkine pruge se ovaj put jasno vide) i ima sposobnost razlikovanja klasa na slici (pokazuje samo mačku, a ne i psa).

Jedna od primjena *Grad-CAM* metode je otkrivanje pristranosti (*bias*-a) modela u trening skupu podataka. Takvi modeli ne generaliziraju situacije stvarnog svijeta ili, još gore, mogu naučiti neke stereotipe na osnovu rase, spola, godina i slično (pogledati [36] i [15] za više detalja). Selvaraju *et al*. su istrenirali konvolucijsku neuronsku mrežu koja sliku klasificira kao "doktor/doktorica" ili kao "medicinska sestra/tehničar". Oko 250 slika (za svaku klasu) prikupili su s jedne poznate internetske tražilice. Podatke su podijelili u trening i testni skup. Na trening skupu je taj CNN model imao dosta veliku točnost, ali na testnom skupu je imao oko 84% točnosti.

Grad-CAM vizualizacija modela je otkrila da je model naučio gledati lice i kosu subjekta na slici kako bi uspio raspoznati radi li se o doktoru/doktorici ili medicinskoj sestri/tehničaru. Naime, kada su pogledali detaljnije u rezultate modela, shvatili su kako je na 78% slika doktora/doktorice bila muška osoba, a na 93% slika medicinske sestre/tehničara je bila ženska osoba. To znači da je model naučio stereotip na osnovu spola što je očito ve-

liki problem. Selvaraju *et al.* su, potaknuti tom informacijom, odlučili ubaciti više slika na kojima je doktorica i više slika na kojima je medicinski tehničar te su zadržali isti broj slika u jednoj i drugoj klasi. Rezultat testnog skupa se povećao na 90%. Na slici 3.5 možemo vidjeti primjere nekih slika iz tog skupa podataka i rezultate modela koji ima pristranost te modela koji nema pristranost. Rezultati tog eksperimenta pokazuju kako *Grad-CAM* može pomoći pri otkrivanju pristranosti modela što je vrlo bitno za generalizaciju, ali i za nediskriminirajuće i etičke rezultate koji su osnovni uvjeti modernog društva.

3.2.2 TCAV - ispitivanje s aktivacijskim vektorima koncepta

Veliki problem metoda koje pronalaze piksele ulazne slike koji najviše utječu na rezultat CNN modela je to što u većini slučajeva ti pikseli ne reprezentiraju značajke/koncepte viših razina koje su ljudima lako razumljivi. Također, aktivacije neurona pojedinih slojeva ostaju nerazumljivi. Ako pogledamo s matematičke strane, CNN model možemo prikazati u vektorskom prostoru E_m određenog vektorima baze e_m koji predstavljaju ulazne vrijednosti i neuronske aktivacije. Ljudi razmišljaju u smislu drugačijeg vektorskog prostora E_h koji je određen vektorima baze e_h koji predstavljaju nepoznate koncepte razumljive ljudima. Ako iz te perspektive promatramo interpretabilnost nekog CNN modela, onda ju možemo prikazati preko funkcije $g\colon E_m \rightarrow E_h$. Ako je g linearna, kažemo da se radi o **linearnoj interpretabilnosti**. Općenito, interpretabilnost ne mora biti savršena funkcija koja objašnjava cijelu domenu E_m niti pokriti sve moguće ljudski razumljive koncepte odnosno cijelu kodomenu E_h. U ovom radu, koncepte vektorskog prostora E_h definiramo skupom slika. Npr. ako želimo definirati koncept "kovrčav", onda ga definiramo skupom slika na kojima se nalazi kovrčava kosa ili slično.

Kim *et al.* su uveli pojam *CAV* vektora (*Concept Activation Vector*). *CAV* je vektor normale nad pravcem koji određuje granicu između klase primjera koji predstavljaju neki koncept i nasumično odabranih slika što možemo vidjeti na slici 3.6. Tu granicu pronalazimo treniranjem linearnog klasifikatora. Najvažniji rezultat ovog rada je metoda linearne interpretabilnosti naziva *TCAV* (*Testing with Concept Activation Vectors*) čiji način rada možemo vidjeti na slici 3.6. *TCAV* metoda koristi takozvanu usmjerenu derivaciju $S_{C,k,l}$ kako bi odredila osjetljivost CNN modela na neki koncept kojeg predstavlja *CAV* vektor. Npr. za dani CNN model koji prepoznaje zebre i za dani skup slika, definiranog od strane korisnika i koje predstavljaju koncept "pruge", *TCAV* metoda može odrediti utjecaj odnosno osjetljivost CNN modela na koncept "pruge".

Jednostavnije rečeno, *TCAV* metoda koristi *CAV* vektor koji pokazuje u smjeru aktivacija pojedinog sloja, gdje je svaki aktivirani primjerima slika koje definiraju određeni koncept. Intuitivno govoreći, ukoliko neku sliku promijenimo tako da izgleda više ili manje kao određeni koncept, zanima nas koliko će to utjecati na sami rezultat CNN modela. Ako će se puno promijeniti, onda to znači da je taj određeni koncept vrlo bitan u predikciji,

(a) Original image (b) Grad-CAM for biased model (c) Grad-CAM for unbiased model

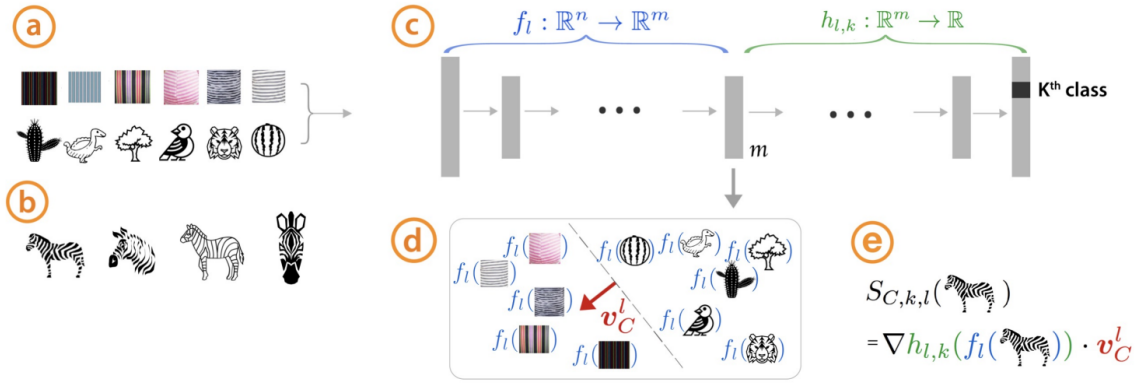
(d) Original Image (e) Grad-CAM for biased model (f) Grad-CAM for unbiased model

(g) Original Image (h) Grad-CAM for biased model (i) Grad-CAM for unbiased model

Slika 3.5: Otkrivanje pristranosti pomoću *Grad-CAM* metode. U prvom redu možemo vidjeti kako su oba modela (onaj s pristranosti i onaj bez pristranosti) uspješno klasificirali sliku a). No, pomoću *Grad-CAM* metode možemo vidjeti kako se model s pristranosti fokusira na lice i kosu medicinske sestre kako bi donio svoju odluku (slika b)), dok se model bez pristranosti fokusira na kratke rukave (slika c)). U drugom i trećem redu vidimo kako je model s pristranosti krivo klasificirao originalnu sliku d) odnosno g)). Opet vidimo kako se taj model fokusirao najviše na lice i kosu promatranog subjekta (slika e) i h)). Model bez pristranosti je dobro klasificirao originalne slike tako da se fokusirao na bijelu kutu i stetoskop. Izvor: [35].

a inače nije. Ovaj proces se ponavlja za velik broj slika neke klase kako bi dobili precizniji rezultat koji se odnosi na osjetljivost klase na neki određeni koncept. Ta osjetljivost se dobije kao omjer broja slika na koje koncept vrlo utječe i svekupnog broja slika na kojima provodimo tu metodu.

Prednost ove metode je što ne zahtijeva veliko znanje strojnog učenja kod korisnika. Također, metoda se prilagođava bilo kojem ljudski razumljivom konceptu te ne zahtijeva dodatno treniranje niti modificiranje promatranog CNN modela.



Slika 3.6: Način rada metode *TCAV*. Ukoliko korisnik metodi preda skup slika koje definiraju neki koncept (npr. "pruge"), nasumične slike koje ne definiraju taj isti koncept (a), označene slike neke određene klase (npr. slike svih zebri u trening skupu) (b) i istreniranu konvolucijsku neuronsku mrežu (c), onda *TCAV* može odrediti osjetljivost modela na određeni koncept za danu klasu. *CAV* vektor se pronalazi učenjem linearnog klasifikatora koji razlikuje aktivacije aktivirane primjerima slika koje predstavljaju određeni koncept te aktivacije aktivirane primjerima nasumičnih slika i to u bilo kojem sloju (d). *CAV* vektor je ortogonalan na pravac koji predstavlja granicu koja razdvaja te dvije klase te je označen s v_C^l crvenom bojom. Kako bi izmjerila osjetljivost neke klase na određeni koncept, *TCAV* metoda se služi takozvanom usmjerenom derivacijom $S_{C,k,l}$. Izvor: [6].

U nastavku objašnjavamo matematičku stranu ove metode. Bez gubitka generalizacije, smatramo da CNN model prima ulaz $x \in \mathbb{R}^n$ te funkcija $f_l \colon \mathbb{R}^n \to \mathbb{R}^m$ označava aktivacije m neurona u l-tom sloju. Za razliku od *saliency* metoda opisanih u Vizualizacija CNN reprezentacije koje kao mjerilo osjetljivosti uzimaju utjecaj piksela ulazne slike, *TCAV* osjetljivost bazira na promjenama ulazne slike u smjeru koncepta određenog *CAV* vektorom u sloju l konvolucijske neuronske mreže. Ako je $v_C^l \in \mathbb{R}^m$ *CAV* vektor za koncept C u sloju l i $f_l(x)$ aktivacija u sloju l za ulaznu sliku x, tada se osjetljivost na koncept (usmjerena

derivacija) C klase k računa kao:

$$S_{C,k,l}(x) = \lim_{\epsilon \to 0} \frac{h_{l,k}(f_l(x) + \epsilon v_C^l) - h_{l,k}(f_l(x))}{\epsilon}, \tag{3.9}$$

gdje je $h_{l,k} : \mathbb{R}^m \to \mathbb{R}$ *logit* funkcija[1].

TCAV metoda koristi usmjerenu derivaciju kako bi izmjerila osjetljivost čitave klase nekog CNN modela nad nekim konceptom. Neka je k dana klasa te neka X_k označava sve slike trening skupa koje pripadaju toj klasi. Definiramo *TCAV* rezultat na sljedeći način:

$$TCAV_{Q_{C,k,l}} = \frac{\left|\{x \in X_k : S_{C,k,l}(x) > 0\}\right|}{|X_k|}, \tag{3.10}$$

gdje je $TCAV_{Q_{C,k,l}} \in [0, 1]$. Treba napomenuti kako u ovom slučaju $TCAV_{Q_{C,k,l}}$ ovisi samo o pozitivnosti $S_{C,k,l}$ što nije nužno za izračun. Korisnik može definirati proizvoljnu metriku koja mu najviše odgovara.

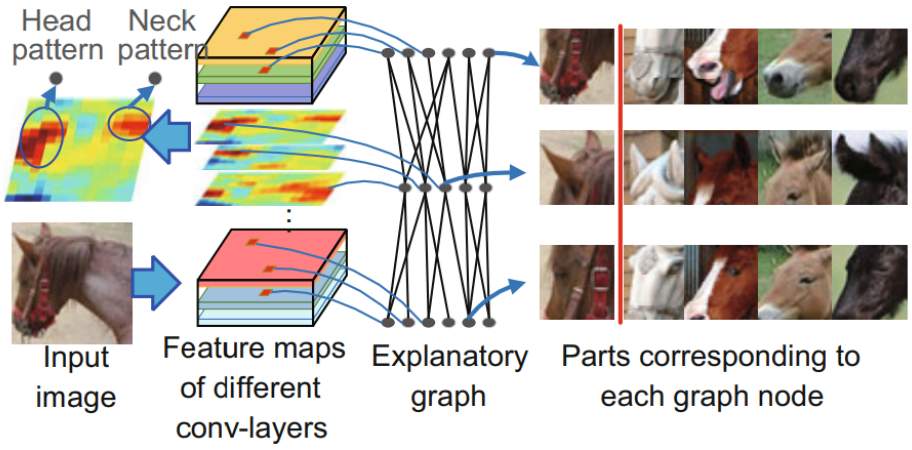
Mana *TCAV* metode je potencijalno loše naučen *CAV* vektor. To se najčešće događa ukoliko korisnik ne priredi dobar skup slika koje definiraju neki koncept i ostalih nasumičnih slika. Kako bi riješili taj potencijalni problem, treniranje se vrši barem kroz 500 epoha te se nakon toga provodi *t-test* kako bi se uvjerili u ispravnost rezultata.

3.3 ”Otpetljavanje” CNN reprezentacije u eksplanatorne grafove

”Otpetljavanje” CNN reprezentacije u ljudski razumljive grafove (eksplanatorne grafove) (slika 3.7) je preciznija i robusnija metoda od vizualizacije i dijagnoze konvolucijskih neuronskih mreža. Zhang *et al.* ([29], [30]) su prvi predložili metodu koja bi konvolucijske slojeve CNN modela ”otpetljala” u grafičke modele koji bi reprezentirali semantičku hijerarhiju skrivenu unutar CNN modela.

Na slici 3.8 možemo vidjeti kako filteri viših konvolucijskih slojeva obično ne predstavljaju samo jedan uzorak nego više njih. Na primjer, filter mogu aktivirati glava i ruke nekog objekta. Kako bi na ispravan način prikazali vizualno znanje koje posjeduje CNN model, cilj Zhang *et al.* je bio odgovoriti na sljedeća tri pitanja:

- Koliko tipova značajki je zapamćeno u svakom filteru konvolucijskog sloja?
- Koje značajke se aktiviraju simultano kako bi opisali dio nekog objekta na slici?
- Kakva je prostorna povezanost[2] takvih dviju značajki?



Slika 3.7: Pojednostavljeni prikaz rada ekplanatornog grafa. Ovdje možemo vidjeti primjer eksplanatornog grafa koji prikazuje vizualno znanje koje posjeduje CNN model. On "otpetljava" izmiješane dijelove uzoraka na svim filterima konvolucijskih slojeva te svakom čvoru u grafu pridružuje jedan dio nekog uzorka. Graf može biti sastavljen od više slojeva te u tom slučaju svaki sloj odgovara jednom konvolucijskom sloju u CNN modelu. Izvor: [29].

Slika 3.8: Mape značajki filtera dobivene iz raznih slika. Možemo vidjeti kako filteri viših konvolucijskih slojeva obično ne predstavljaju samo jedan uzorak nego više njih. Izvor: [29].

Eksplanatorni graf ima isti broj slojeva koliko konvolucijskih slojeva ima jedan promatrani CNN model. Svaki sloj grafa sadržava broj čvorova koji je jednak ili veći od broja filtera u konvolucijskom sloju. Razlog tome je što jedan filter može rezultirati mapom značajki na kojoj se nalazi jedan ili više uzoraka nekog objekta, a čvor u grafu treba reprezentirati svaki mogući uzorak koji se pojavljuje na mapi značajki. Cilj je da svaki čvor reprezentira isti uzorak na bilo kojoj ulaznoj slici što se može vidjeti na slici 3.9. Bridovi grafa predstavljaju veze čvorova susjednih slojeva koji mogu biti povezani zbog prostorne povezanosti ili zbog toga što se oba čvora u većini slučajeva skupa aktiviraju[3].

Slika 3.9: Vizualni prikaz čvorova eksplanatornog grafa. Očito je kako svaki od četiri prikazana čvora prikazuje isti uzorak na različitim slikama što je i cilj eksplanatornog grafa. Izvor: [29].

Treba napomenuti da se pozicija čvora ne fiksira u eksplanatornom grafu jer se uzorci mogu pojaviti na bilo kojoj poziciji na nekoj mapi značajki. Npr. na slici 3.7 možemo vidjeti kako se uzorak konjskog uha i lica može pojaviti na različitim pozicijama sve dok su na isti način prostorno povezani te se zajedno aktiviraju.

Eksplanatorni graf se pronalazi jednostavnom metodom preko nenadziranog učenja što znači da nema potrebe za dodatnom anotacijom podataka. To je vrlo korisno kod lokalizacije objekata na slici te se pokazalo da čvorovi eksplanatnorog grafa nadmašuju neke druge metode lokalizacije koje zahtjevaju nadzirano učenje. Također, eksperimenti su pokazali da čvorovi eksplanatornog grafa ispunjuju svoj cilj reprezentiranja istog uzorka na različitim slikama što ovu metodu čini veoma popularnom.

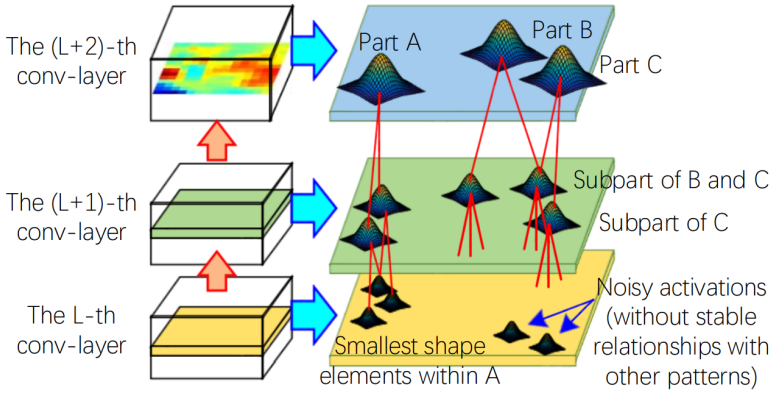
Kao što se može vidjeti na slici 3.10, filter može biti aktiviran raznim oblicima promatranog objekta i to na raznim mjestima. Neki od tih aktiviranih oblika predstavljaju

[1] $h_{l,k}(x) = \log \frac{x}{1-x}$.

[2] Povezanost odnosno udaljenost na osnovu piksela neke slike.

[3] Čvor ima jaku aktivaciju ako uzorak kojeg predstavlja ima visok utjecaj na rezultat konvolucijske neuronske mreže.

uzorke koji odgovaraju nekom dijelu promatranog objekta pa takve uzorke nazivamo *djelomični uzorci*. Ostali uzorci predstavljaju šum. Cilj je pronaći razliku između *djelomičnih uzoraka* i uzoraka koji predstavljaju šum. Pretpostavlja se da ako aktivirani oblik na mapi značajki jednog filtera predstavlja *djelomični uzorak* onda se mora aktivirati i oblik na mapi značajki drugog filtera na istoj ili vrlo bliskoj poziciji i obrnuto. Ti uzorci predstavljaju dijelove nekog uzorka u kasnijem konvolucijskom sloju. Upravo zbog toga, u eksplanatornom grafu, bridovima povezujemo susjedne slojeve. Pronalaženje čvorova (uzoraka) se odvija od kasnijih prema ranijim konvolucijskim slojevima prolazeći kroz slike iz skupa podataka korištenog za treniranje. Promatraju se čvorovi koji imaju najstabilnije prostorne povezanosti i koji se skupa aktiviraju. Ako promotrimo sliku 3.10, vidimo kako se u kasnijim konvolucijskim slojevima pojavljuju aktivacije veće površine koji predstavljaju veće uzorke na promatranom objektu dok se u ranijim konvolucijskim slojevima pojavljuju aktivacije manjih površina koje većinom ne možemo smjestiti u semantičku strukturu. Upravo te aktivacije kasnijih konvolucijskih slojeva pomažu pri odbacivanju šumova u ranijim konvolucijskim slojevima.



Slika 3.10: Prikaz prostorne povezanosti i aktivacije nekih dijelova uzoraka. Vidljivo je kako uzorci kasnijih konvolucijskih slojeva zanemaruju šumove koji se pojavljuju u ranijim konvolucijskim slojevima. Ako pogledamo iz drugačije perspektive, dijelove uzoraka ranijih konvolucijskih slojeva možemo shvaćati kao komponente uzoraka koji se pojavljuju u kasnijim konvolucijskim slojevima. Izvor: [29].

Kao što smo spomenuli, eksplanatorni graf G se gradi od kasnijih prema ranijim konvolucijskim slojevima. Prvo se, gledajući sve slike trening skupa $I \in \mathbf{I}$, promatra zadnji konvolucijski sloj i na osnovu njega se kreiraju čvorovi zadnjeg sloja grafa G. Ti se čvorovi/uzorci koriste pri "otpetljavanju" nejasnih aktiviranih oblika na mapama značajki predzadnjeg konvolucijskog sloja te se postupak ponavlja prema ranijim konvolucijskim

slojevima. Konkretno, za dobivene pozicije uzoraka $\{R^I_{L+1}\}_{I\in\mathbf{I}}$ koje predstavljaju čvorovi u $L+1$-om sloju eksplanatornog grafa, se očekuje da se vrlo dobro poklapaju s aktivacijama mapi značajki L-tog konvolucijskog sloja. Također, od prepoznatih uzoraka L-tog konvolucijskog sloja se očekuje da održavaju dobru prostornu povezanost sa uzorcima $L+1$-og konvolucijskog sloja koje smo već reprezentirali preko čvorova eksplanatornog grafa. Funkcija cilja kojom dobivamo L-ti sloj eksplanatornog grafa je sljedeća:

$$\underset{\theta_L}{\operatorname{argmax}} \prod_{I\in\mathbf{I}} P(X^I_L|R^I_{L+1},\theta_L), \tag{3.11}$$

gdje je X^I_L skup mapi značajki L-tog konvolucijskog sloja dobivenih slikom I, a θ_L su parametri ekplanatornog grafa koje je potrebno naučiti.

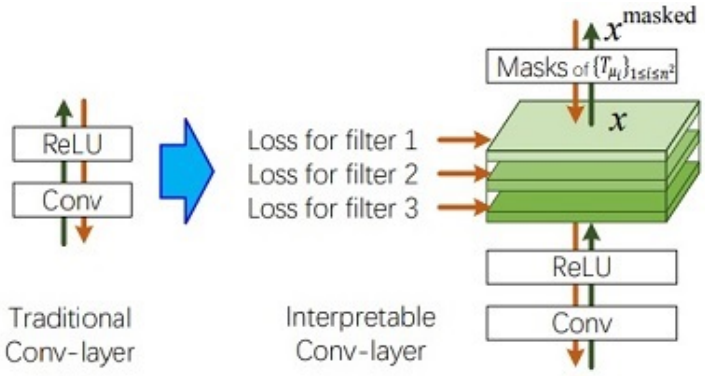
3.4 Učenje neuronske mreže koja ima interpretabilnu reprezentaciju

Sve metode koje smo do sada pokazali pokušavaju što bolje razumjeti već istreniranu konvolucijsku neuronsku mrežu. Ipak, postoji mali broj radova koji je orijentiran na treniranje konvolucijske neuronske mreže s reprezentacijom koja ima jasno semantičko značenje što je velik izazov za *end-to-end* učenje.

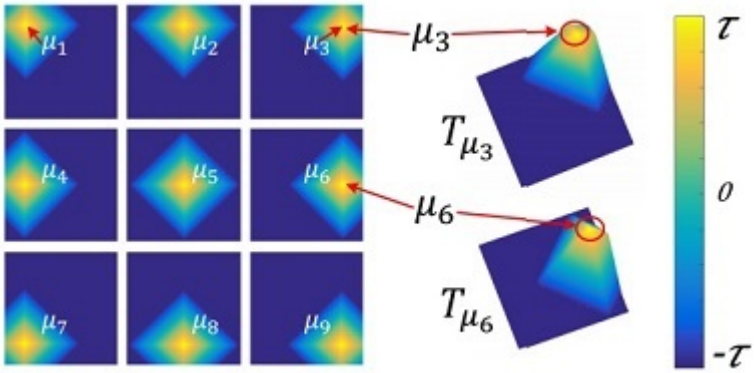
Zhang *et al.* ([33]) su razvili metodu koja neznatno mijenja arhitekturu postojeće konvolucijske neuronske mreže, ali značajno povećava njenu interpretabilnost. Ne mijenja se njena funkcija gubitka te se CNN model trenira nad istim podacima bez dodatnog anotiranja istih. Jedino što dodajemo arhitekturi CNN modela je funkcija gubitka za svaki filter što možemo vidjeti na slici 3.11. Ta funkcija je uvedena kako bi mape značajki reprezentirale neki specifični dio objekta kojeg promatramo, a ne više njih kako je to inače slučaj. Zhang *et al.* su pretpostavili da ako filter reprezentira više regija slike, onda vjerojatno opisuje neke značajke niže razine (npr. rubovi i boje), umjesto značajki viših razina što je nužno za interpretabilnost samog CNN modela.

Funkcija gubitka automatski dodjeljuje svakom filteru neki dio promatranog objekta u *end-to-end* procesu učenja. To se postiže određenim uzorcima T_{μ_i} kao što možemo vidjeti na slici 3.12. Ukoliko je mapa značajki nekog filtera dimenzija $n\times n$, onda se za taj filter konstruira n^2 različitih uzoraka T_{μ_i} također dimenzija $n\times n$. Razlog tolikom broju uzoraka T_{μ_i} je mogućnost pojave dijela promatranog objekta na svakog dijelu slike. Za svaki uzorak T_{μ_i} vrijedi $T_{\mu_i} = t^+_{i,j}$ gdje je $t^+_{i,j} = \tau\cdot\max\{1-\beta\frac{\|[i,j]-\mu\|_1}{n}, -1\}$ gdje je $\|\cdot\|_1$ 1-norma, a β je konstanta koju sami određujemo.

Neka $\mathbf{I}$ označava trening skup, a I_c neka označava sliku koja pripada klasi c. Za svaku sliku $I\in\mathbf{I}$ tijekom unaprijedne faze CNN model odabire određeni uzorak $T_{\hat{\mu}}$ između n^2 ostalih uzoraka kao masku koja će zanemariti šumove na ostalim mjestima mape značajki



Slika 3.11: Struktura tradicionalnog i interpretabilnog konvolucijskog sloja. S lijeve strane je prikazana tradicionalna struktura konvolucijskog sloja s *ReLu* aktivacijskom funkcijom dok u interpretabilnom konvolucijskom sloju, kojeg su uveli Zhang *et al.* ([33]), vidimo prisutnu funkciju gubitka zajedno sa uzorcima T_{μ_i}. Izvor: [33].



Slika 3.12: Uzorci osmišljeni od strane Zhang *et al.* ([33]). Idealna situacija koja se želi postići je da svaka mapa značajke izgleda kao jedan od uzoraka T_{μ_i}. Ukoliko je veličina mape značajke nekog filtera $n \times n$, tada imamo n^2 takvih uzoraka. Razlog tolikom broju uzoraka T_{μ_i} je mogućnost pojave dijela promatranog objekta na svakog dijelu slike. Konstantu τ određujemo sami. Izvor: [33].

x nakon *ReLu* aktivacijske funkcije. Prvo računamo $\hat{\mu} = \text{argmax}_{[i,j]}\, x_{i,j}$ i $x^{\textbf{masked}} = \max\{x \odot T_{\hat{\mu}}, 0\}$ gdje je $\odot$ operator množenja *element po element*. $\hat{\mu}$ predstavlja potencijalnu središnju lokaciju dijela promatranog objekta. Maskirana mapa značajki $x^{\textbf{masked}}$ sudjeluje u algoritmu s povratnim postupkom. Treba napomenuti da se za svaku sliku I mogu izabrati

različiti uzorci $T_{\hat{\mu}}$ što se može vidjeti na slici 3.13.

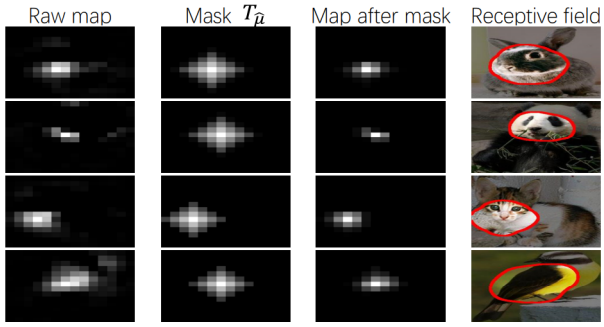
Tijekom algoritma s povratnim postupkom funkcija gubitka, koju ćemo spomenuti kasnije, navodi filter f da se aktivira samo na jednom specifičnom dijelu promatranog objekta klase c te da se uopće ne aktivira u slučaju klase koja nije c. Neka je $\mathbf{X} = \{x | x = f(I), I \in \mathbf{I}\}$ skup mapi značajki filtera f nakon *ReLu* aktivacijske funkcije dobivenih računanjem na različitim slikama. Ako I pripada klasi c, onda se očekuje da mapa značajki izgleda kao uzorak $T_{\hat{\mu}}$, u suprotnom slučaju očekujemo da mapa značajki odgovara negativnom uzorku $T^- = (t^-_{i,j})$, gdje je $t^-_{i,j} = -\tau < 0$ i τ neka pozitivna konstanta. Podsjetimo kako tijekom unaprijedne faze nije moguće izabrati negativni uzorak. Očekujemo da svaka mapa značajki bude ista ili barem slična jednom od uzoraka $\mathbf{T} = \{T^-, T_{\mu_1}, T_{\mu_2}, ..., T_{\mu_{n^2}}\}$ nakon *ReLu* aktivacijske funkcije. Funkcija gubitka je zamišljena kao međusobna informacija između $\mathbf{X}$ i $\mathbf{T}$ (eng. *mutual information*) te se definira na sljedeći način:

$$\mathbf{Loss}_f = -MI(\mathbf{X}; \mathbf{T}) = -\sum_T p(T) \sum_x p(x|T) \log \frac{p(x|T)}{p(x)}, \tag{3.12}$$

gdje su $p(T_\mu) = \frac{\alpha}{n^2}$ i $p(T^-) = 1 - \alpha$ gdje je α konstanta koju sami određujemo. $p(x|T)$ određujemo na sljedeći način:

$$\forall T \in \mathbf{T}, p(x|T) = \frac{1}{Z_T} \exp[tr(x \cdot T)], \tag{3.13}$$

gdje je $Z_T = \sum_{x \in \mathbf{X}} \exp(tr(x \cdot T))$ a tr trag matrice. Naposljetku, imamo $p(x) = \sum_T p(T)p(x|T)$.



Slika 3.13: Prikaz različitih uzoraka $T_{\hat{\mu}}$ za različite slike. Stupci redom prikazuju mape značajki nakon *ReLu* aktivacijske funkcije, uzorak $T_{\hat{\mu}}$ koji ima ulogu maske, rezultat njihovog množenja $x^{\mathbf{masked}}$ te vizualizaciju na ulaznoj slici I. Svaki red prikazuje drugu ulaznu sliku I. Izvor: [33].

3.5 Interpretabilnost *middle-to-end* učenja

U ovom potpoglavlju se bavimo istraživanjem vezanim uz *middle-to-end* učenje konvolucijskih neuronskih mreža. Zhang *et al.* ([30]) su predložili metodu koja koristi *question-answering* i AOG (*And-Or*) grafove u svrhu boljeg semantičkog razumijevanja CNN reprezentacije. Model radi nad već istreniranom konvolucijskom neuronskom mrežom i gradi AOG grafove na osnovu mapi značajki konvolucijskih slojeva. Naučeni AOG graf možemo iskoristiti kod lokalizacije dijelova promatranog objekta, a sastoji se od tri vrste čvorova. Prva vrsta su *OR* čvorovi koji predstavljaju reprezentacije koje tvore neki dio promatranog objekta (npr. glava, ruke i noge), dok *AND* čvorovi predstavljaju reprezentacije koje skupa tvore sastavni dio nekog dijela promatranog objekta. *Terminal* čvorovi predstavljaju mape značajki konvolucijskih slojeva. AOG graf se sastoji od četiri sloja:

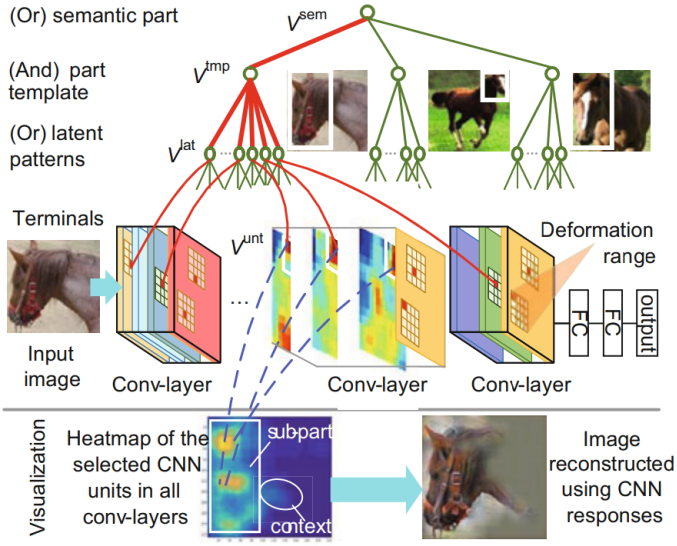
- **semantički sloj** – *OR* čvorovi ovog sloja predstavljaju semantički dio nekog objekta poput glave konja kao što možemo vidjeti na slici 3.14
- **sloj djelomičnih uzoraka** – *AND* čvorovi ovog sloja su djeca nekog čvora u semantičkom sloju. Predstavljaju dijelove objekta bez obzira na poziciju ili kut gledanja. Na slici 3.14 možemo vidjeti kako čvorovi predstavljaju glavu konja u različitim pozicijama, pod različitim kutom i drugačijeg izgleda
- **sloj skrivenih uzoraka** – *OR* čvorovi ovog sloja predstavljaju, na prvi pogled, nejasne dijelove nekog objekta poput ruba nosa ili vrata
- **krajnji sloj** – *Terminal* čvorovi ovog sloja su mape značajki konvolucijskih slojeva. Njihov čvor roditelj je čvor iz sloja skrivenih uzoraka kojem te mape značajki odgovaraju njegovoj reprezentaciji.

Treba napomenuti da je za ovu metodu potrebno dodatno anotirati semantičke dijelove manjeg broja slika u trening skupu. Obično se anotiraju dijelovi objekta na tri do 12 slika iz trening skupa koji predstavljaju najvažnije uzorke kako bi se skrivenih uzorci lakše pronašli.

Ilustracija je prikazana na slici 3.14. Osnovni cilj ove metode je dobiti metriku koja razlikuje skrivene uzorke od šumova u konvolucijskim slojevima. Od skrivenih uzoraka se očekuje da konzistentno predstavljaju određeni dio nekog objekta koji je anotiran od strane autora samog modela, da frenkventno predstavljaju dijelove objekta koji nije anotiran te da sa ostalim skrivenim uzorcima međusobno održavaju prostornu povezanost.

Ova metoda je vrlo dobra za lokalizaciju dijelova objekta na slici. Naime, ona uz slabo nadzirano (eng. *weakly supervised*) učenje (anotiranje samo 3-12 slika) ima mogućnost unaprijediti performanse već istrenirane konvolucijske neuronske mreže za 13% do 107%.

Prije nego objasnimo parsiranje, pogledajmo notaciju u tablici 3.1 u kojoj su prikazane notacije čvorova. Postoje dva slučaja parsiranja. Ukoliko imamo aktivacijsku mapu koja



Slika 3.14: AOG (*And-Or*) semantički graf nad već istreniranom konvolucijskom neuronskom mrežom. Na ovom primjeru je prikazan samo jedan čvor semantičkog sloja (prvi red) koji predstavlja glavu konja. Ispod njega dolazi sloj djelomičnih uzoraka koji prikazuje glavu konja u raznim pozicijama gdje veličina i izgled glave konja mogu varirati. Nakon njega dolazi sloj skrivenih uzoraka koji predstavlja ne tako jasne uzorke i njih je najteže razlikovati od šumova. Krajnji sloj se oblikuje preko svog čvora roditelja, a zapravo se sastoji od mapi značajki konvolucijskih slojeva. U zadnjem redu možemo vidjeti vizualizaciju gdje su se aktivirale pojedine mape značajki koje odgovoraju glavi konja. Izvor: [31]

pokazuje željeni semantički dio, parsiranje se vrši ”od vrha prema dnu” prolazeći kroz AOG graf. Prvo se odredi djelomičan uzorak V^{tmp} koji je *dijete čvor* semantičkog uzorka V^{sem} kojeg tražimo. Nakon toga se parsira regija slike na kojoj je taj djelomičan uzorak te se nakon toga gleda svako njegovo dijete V^{lat} koje određuje mapu značajki V^{unt} nekog konvolucijskog sloja s blagom deformacijom svoje pozicije na originalnoj slici. Na ovaj način odabiremo određene čvorove AOG grafa koji opisuju neke dijelove promatranog objekta što je prikazano crvenim linijama na slici 3.15. Kažemo da ti čvorovi pripadaju grafu parsiranja. Pretpostavka je da CNN model radi dovoljno dobro da može lokalizirati semantički uzorak s kojim krećemo raditi parsiranje.

Ukoliko imamo ulaznu sliku I i pripadni AOG graf, parsiranje računamo dinamičkim programiranjem ”od dna prema vrhu”. Napomenimo da P_V označava centralnu poziciju neke regije Λ_V na ulaznoj slici, a $\overline{P}_V$ označava ”idealnu” centralnu poziciju neke regije za

sloj	**ime**	**vrsta čvora**	**notacija**
1	Semantički sloj	OR	V^{sem}
2	Sloj djelomicnih uzoraka	AND	$V^{\text{tmp}} \in \Omega^{\text{tmp}}$
3	Sloj skrivenih uzoraka	OR	$V^{\text{lat}} \in \Omega^{\text{lat}}$
4	Krajnji sloj	Terminal	$V^{\text{unt}} \in \Omega^{\text{unt}}$

Tablica 3.1: Notacija čvorova pojedinih slojeva.

razliku od deformirane koju smo spomenuli.

Terminalni čvorovi (u krajnjem sloju): Svakom čvoru V^{unt} pridružujemo fiksnu regiju slike $\Lambda_{V^{\text{unt}}}$ te računamo:

$$S_I(V^{\text{unt}}) = S_I^{\text{rsp}}(V^{\text{unt}}) + S_I^{\text{loc}}(V^{\text{unt}}) + S_I^{\text{pair}}(V^{\text{unt}}), \tag{3.14}$$

gdje su:

$$S_I^{\text{rsp}}(V^{\text{unt}}) = \begin{cases} \lambda^{\text{rsp}} X(V^{\text{unt}}), & X(V^{\text{unt}}) > 0 \\ \lambda^{\text{rsp}} S_{\text{none}}, & X(V^{\text{unt}}) \leq 0 \end{cases} \tag{3.15}$$

$$S_I^{\text{pair}}(V^{\text{unt}}) = -\lambda^{\text{pair}} \text{mean}_{V^{\text{lat}}_{\text{upper}} \in Neighbor(V^{\text{lat}})} \left\| [P_{V^{\text{unt}}} - \overline{P}_{V^{\text{lat}}_{\text{upper}}}] - [\overline{P}_{V^{\text{lat}}_{\text{upper}}} \overline{P}_{V^{\text{lat}}}] \right\|. \tag{3.16}$$

Rezultat $S_I(V^{\text{unt}})$ se sastoji od tri dijela:

1) $S_I^{\text{rsp}}(V^{\text{unt}})$ označava aktivaciju mape značajki V^{unt} kada je slika I na ulazu konvolucijske neuronske mreže. $X(V^{\text{unt}})$ označava normaliziranu aktivaciju V^{unt} dok $S_{\text{none}} = -3$ označava neaktiviranu mapu značajki.

2) Jednom kada čvor roditelj V^{lat} izabere dijete V^{unt}, $S_I^{\text{loc}}(V^{\text{unt}})$ mjeri udaljenost deformirane lokacije $P_{V^{\text{unt}}}$ i "idealne" lokacije $\overline{P}_{V^{\text{lat}}}$.

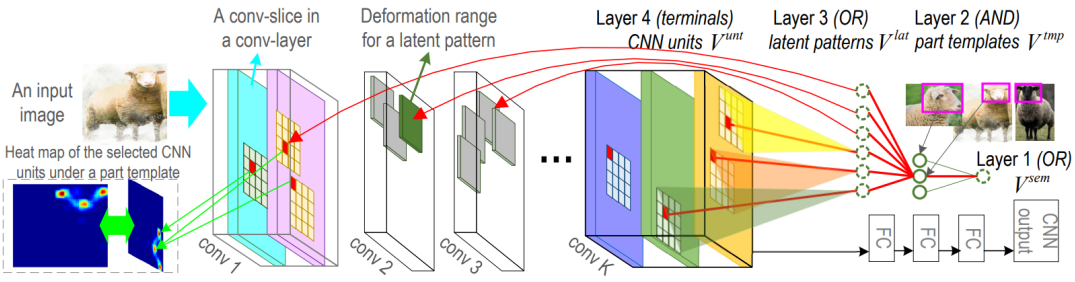
3) $S_I^{\text{pair}}(V^{\text{unt}})$ mjeri prostornu povezanost susjednih skrivenih uzoraka.

Konstante λ^{rsp} i λ^{pair} postavljamo na 1.5 odnosno 10.0.

Čvorovi sloja skrivenih uzoraka: Svaki čvor V^{lat} je određen svojom djecom odnosno mapama značajki čiju smo poziciju namjerno deformirali količinom deformacije koju određujemo sami. Na osnovu rezultata $S_I(V^{\text{unt}})$ koje smo izračunali, biramo dijete V^{unt} na osnovu:

$$S_I^{(}V^{\text{lat}}) = \max_{V^{\text{unt}} \in Child(V^{\text{lat}})} S_I(V^{\text{unt}}), \qquad \Lambda_{V^{\text{lat}}} = \Lambda_{V^{\text{unt}}}. \tag{3.17}$$

Čvorovi sloja djelomičnih uzoraka: Svaki čvor V^{tmp} je *AND* čvor koji ovisi o rezultatu $S_I(V^{\text{lat}})$ svoje djece. Računamo regiju slike $\Lambda_{V^{\text{tmp}}}$ koja maksimizira sljedeći rezultat:



Slika 3.15: Ilustracija metode. Izvor: [30]

$$S_I(V^{\text{tmp}}) = \max_{\Lambda_{V^{\text{tmp}}}} \sum_{V^{\text{lat}} \in Child(V^{\text{tmp}})} [S_I(V^{\text{lat}}) + S_I^{\text{inf}}(\Lambda_{V^{\text{tmp}}}, \Lambda_{V^{\text{lat}}})] \tag{3.18}$$

gdje je $S_I^{\text{inf}}(\Lambda_{V^{\text{tmp}}}, \Lambda_{V^{\text{lat}}})$ funkcija koja mjeri prostornu povezanost između V^{tmp} i svakog njegovog djeteta V^{lat}.

Čvorovi semantičkog sloja: Svaki čvor V^{sem} računamo na sličan način kao i čvorove sloja skrivenih uzoraka V^{lat}. Računamo:

$$S_I^{(}V^{\text{sem}}) = \max_{V^{\text{tmp}} \in Child(V^{\text{sem}})} S_I(V^{\text{tmp}}), \qquad \Lambda_{V^{\text{sem}}} = \Lambda_{V^{\text{tmp}}}. \tag{3.19}$$

Poglavlje 4

Eksperimentalna evaluacija odabranih metoda interpretabilnosti

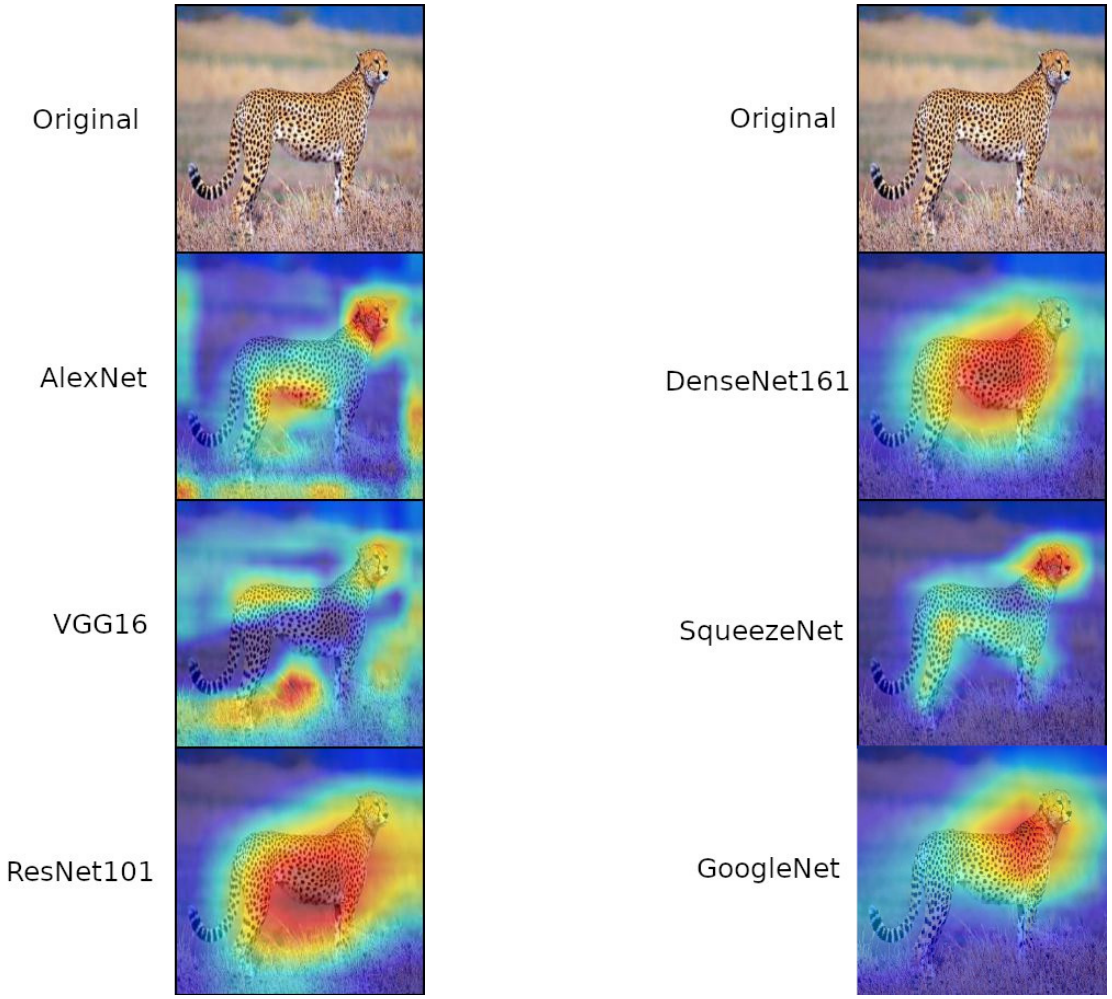
U ovom poglavlju ćemo pokazati primjene *Grad-CAM* i *TCAV* metode nad nekim konkretnim primjerima. Cilj nam je pobliže se upoznati s tim metodama i prokomentirati njihove rezultate. Pošto je vrlo teško osmisliti metriku interpretabilnosti, vodit ćemo se ljudskim razumom i intuicijom te pokušati objasniti neke rezultate.

4.1 Grad-CAM

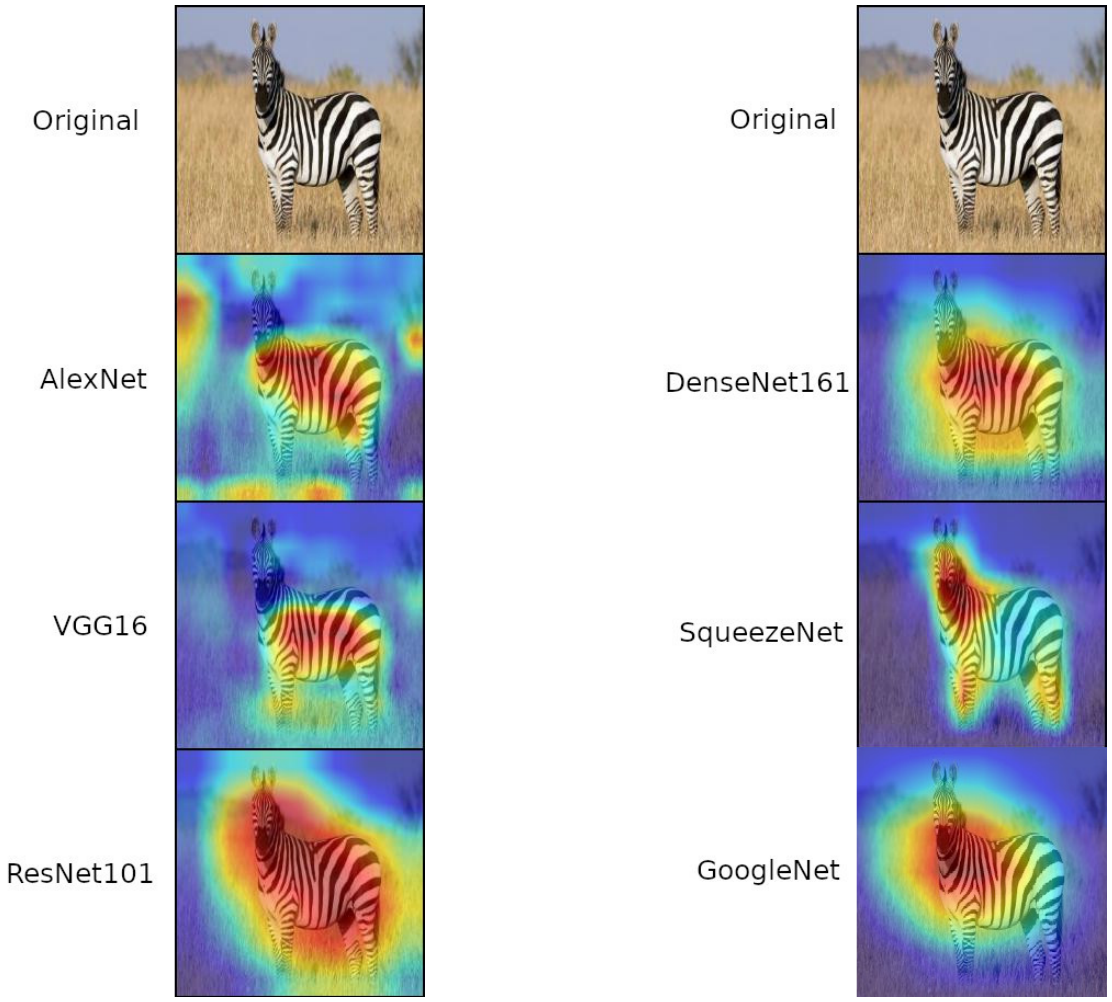
Ovu metodu smo već opisali u poglavlju Dijagnoza CNN reprezentacije i sada ćemo ju pokazati na par primjera. Implementaciju ove metode smo napravili u programskom jeziku *Python* odnosno preciznije u *Pytorch*-u koji je biblioteka u *Python*-u s alatima korisnima u strojnom učenju. *Pytorch* je, između ostaloga, poznat po tome što omogućava izvođenje svog koda na grafičkoj kartici na vrlo jednostavan način. Kodovi za ovu metodu su preuzeti s dva *GitHub* repozitorija - [38] i [22]. Važno je napomenuti da su ti kodovi mijenjani od strane autora odnosno da je autor prilagodio te kodove svojim potrebama i popravio neke dijelove koda koji su mu predstavljali problem u ovom zadatku.

U svrhu boljeg upoznavanje ove metode, pogledat ćemo tri primjera. Radi se o slikama geparda, zebre i bicikla za dvoje osoba. Naime, CNN modeli koje koristimo mogu prepoznati 1000 različitih klasa te između ostaloga i klasu geparda, zebri i bicikala za dvoje. Za svaku od tih slika pokazat ćemo pripadne lokalizacijske mape konvolucijskih neuronskih mreža koje smo spomenuli u potpoglavlju Primjeri konvolucijskih neuronskih mreža. Slike prikazujemo u dva stupca na kojoj se vide lokalizacijske mape za svaki CNN model.

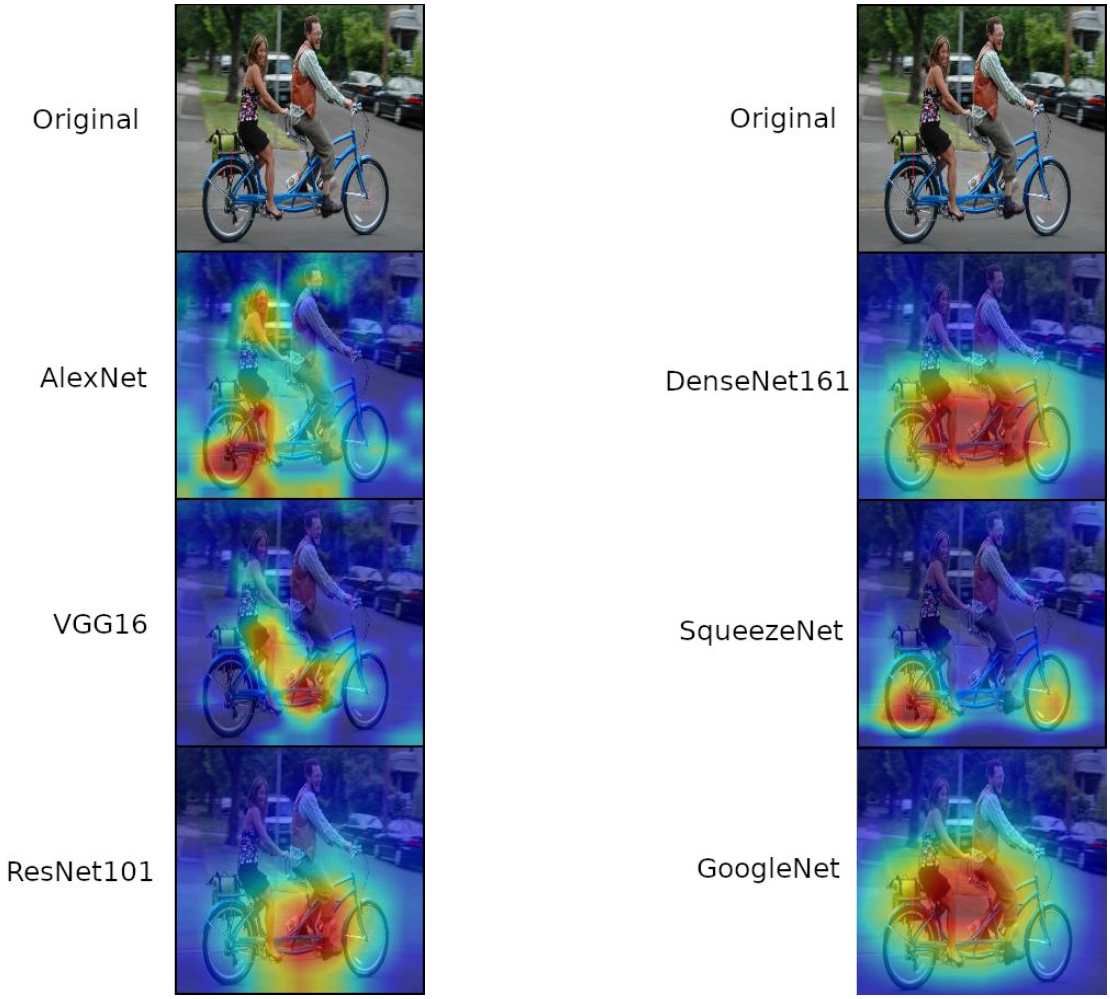
Iz prethodnih primjera možemo zaključiti par stvari. *AlexNet* je definitivno najlošija mreža po pitanju predikcije jer se ne fokusira na karakteristične dijelove objekta koji pripada nekoj klasi. *VGG16* također nema jasnu interpretabilnost, ali za te dvije mreže smo



Slika 4.1: Prikaz lokalizacijskih mapi raznih CNN modela nad slikom geparda. Najuočljivije karakteristike geparda su crne točkice koje se nalaze na tijelu i specifično dugom repu. Također, gepard ima vrlo malu glavu naspram svog tijela. U ovom primjeru možemo vidjeti kako se *AlexNet* i *VGG16* mreža fokusiraju na dosta stvari oko samog geparda što nije dobro. To znači da možemo na neki način promijeniti piksele u crvenom dijelu slike i rezultat će se nemalo promijeniti. No, interpretabilnost ta dva modela odgovara njihovim performansama u smislu točnosti nad testnim skupom podataka. *ResNet101* i *DenseNet161* imaju sličnu lokalizacijsku mapu, ali *ResNet101* je malo manje precizan u ovom slučaju. Zanimljivo je da je *SqueezeNet* jedina mreža koja se fokusira isključivo na geparda, osim njegovog repa. Glava igra najveću ulogu u donošenju predikcije što je vrlo jasno. *GoogleNet* je jedina mreža koja u obzir uzima i cijeli rep geparda, ali se najviše fokusira na njegov vrat u donošenju predikcije. To je sasvim razumno jer je gepardov vrat specifičan pošto spaja snažno i veliko tijelo s malom glavom.



Slika 4.2: Prikaz lokalizacijskih mapi raznih CNN modela nad slikom zebre. Najoučljivija karakteristika zebre su crno-bijele pruge te očekujemo da se CNN modeli najviše na to fokusiraju. Na ovom primjeru vidimo kako se uistinu većina modela fokusira na pruge. No, kod *AlexNet* modela vidimo opet dosta nejasnu lokalizacijsku mapu koja gleda i ostale stvari osim same zebre, dok *VGG16*, u ovom slučaju, ima dobru lokalizacijsku mapu, osim što se uopće ne fokusira na glavu zebre. *ResNet101* i *DenseNet161* ponovno imaju sličnu lokalizacijsku mapu i ponovno je *ResNet101* malo manje precizan nego *DenseNet161*. *SqueezeNet* u ovom slučaju ima skoro pa idealnu lokalizacijsku mapu jer se fokusira samo na zebrino tijelo. Najviše gleda glavu, vrat i noge. *GoogleNet* opet najviše gleda vrat životinje kao i na slici 4.1.



Slika 4.3: Prikaz lokalizacijskih mapi raznih CNN modela nad slikom bicikla za dvoje osoba. Ovo je primjer u kojemu se najviše razlikuju lokalizacijske mape CNN modela s kojima raspolažemo. Bicikl za dvoje ima karakteristike dosta razmaknutih kotača, specifičan srednji dio bicikla i dva sjedala (na kojima mogu biti osobe). *AlexNet* je definitivno loš u ovom slučaju. Naime, fokusira se najviše na stražnji kotač, a prednji ni ne gleda. Fokusira se i na lice osoba koje se voze, ali to nije dobro u slučaju da osobe nisu na biciklu. *VGG16* gleda srednji dio bicikla i stražnje sjedalo što može biti loše jer se volan u ovom slučaju ne gleda. Isto tako je ovo mogao biti bickl za jednu osobu i mreža bi vjerojatno to krivo klasificirala. Zanimljivo je da u ovom primjeru *ResNet101* ima manje crvene boje na lokalizacijskom mapi nego *DenseNet161* jer to nije bio slučaj u prošlim primjeru. No, opet se vidi kako *DenseNet161* gleda konkretnije područje. *SqueezeNet* se opet fokusira na specifična područja slike (ima "urednu" lokalizacijsku mapu) i to na dva kotača. S obzirom da mreža na razne načine može shvatiti udaljenost tih dvaju kotača, može se reći da *SqueezeNet* na logičan način prepoznaje bicikl za dvije osobe. *GoogleNet* se u ovom slučaju fokusira samo većinom na srednji dio bicikla odnosno dva sjedala i nema šumova sa strane koji utječu na predikciju.

i očekivali ovakav rezultat jer nisu najbolje po pitanju perfomansi. Zanimljivo je da su *SqueezeNet* i *GoogleNet* "lakše" varijante *AlexNet* CNN modela, ali ostvaruju bolje performanse te imaju jasniju interpretabilnost nego *AlexNet*. Iz priloženih slika se vidi kako *SqueezeNet* ima nešto jasniju i precizniju interpretabilnost nego *GoogleNet*, ali i od ostalih konvolucijskih neuronskih mreže koje imaju puno veću strukturu nego ona. S obzirom na odlične perfomanse i prikazanu interpretabilnost, može se reći da su *ResNet101* i *DenseNet161* CNN modeli s najboljom učinkovitosti od svih prikazanih. Izdvojio bih ipak *DenseNet161* kao CNN model koji ima jasniju reprezentaciju nego *ResNet101* što se i očekivalo s obzirom da je *DenseNet161* zamišljen kao nekakvo unaprijeđenje *ResNet101* CNN modela.

Ono što je bitno je da ovaj pristup može dati puno informacija osobi koja planira izabrati jedan od ovih modela u svoje svrhe. To što *AlexNet* i *SqueezeNet* imaju sličnu točnost na *ImageNet* skupu podataka, ne znači da su podjednako dobre što možemo i vidjeti iz prethodnih primjera.

Sljedeće što ćemo pogledati su dva primjera loše klasifikacije slika te njihovu pripadnu lokalizacijsku mapu. Svi rezultati su dobiveni *GoogleNet* konvolucijskom neuronskom mrežom. Za svaku od tih slika ćemo pokazati i najveće postotke za neke predviđene klase.

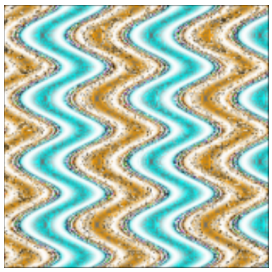
Na slici 4.4 vidimo kraljicu Elizabetu II. Naime, *GoogleNet* ovu sliku prepoznaje kao "kapa za tuširanje" te s desne strane možemo vidjeti pripadnu lokalizacijsku mapu. Očito je kako se ovaj CNN model fokusirao na nebitne značajke odnosno dijelove ove slike. Zanimljivo je da je model klasificirao ovu sliku kao "kapa za tuširanje" sa sigurnošću od 89.84%, dok je s 4.52% klasificirao ovu sliku kao "perika".

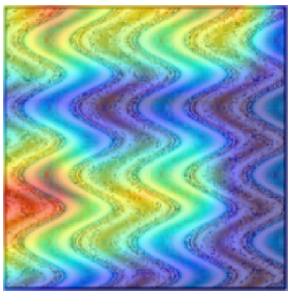
Na slici 4.5 vidimo sliku koju ni ljudska osoba ne može klasificirati, ali *GoogleNet* ju je uspio klasificirati sa sigurnošću 80.15% i to kao "labirint" te kao "moždani koralj" sa sigurnošću od 13.86%. S desne strane vidimo lokalizacijsku mapu koja nam ne otkriva razumna objašnjenja.

4.2 TCAV - ispitivanje s aktivacijskim vektorima koncepta

Metodu *TCAV* smo već spomenuli u poglavlju Dijagnoza CNN reprezentacije, a sad ćemo ju pokazati na jednom primjeru. Promatramo koliko koncepti "cik-cak pruge", "točkice" i "ravne pruge" utječu na predikciju *GoogleNet* CNN modela na klasu "zebra". Koristimo 30 slika koje definiraju spomenute koncepte (po deset za svaki koncept). Te slike možemo vidjeti na slici 4.6. Koristimo i 20 nasumično odabranih slika, nađenih pomoću *Google* tražilice, koje zajedno sa slikama koncepata služe treniranju linearnog klasifikatora. Također, koristimo i 15 slika zebre koje predstavljaju navedenu klasu. Koristili smo kod s *GitHub*-a napisan od strane samih autora članka u kojem je predstavljena spomenuta

Slika 4.4: Slika Elizabete II., kraljice Ujedinjenog kraljevstva i pripadna lokalizacijska mapa *GoogleNet* konvolucijske neuronske mreže koja je napravila predikciju ove slike. Naime, *GoogleNet* CNN model je ovu sliku klasificirao kao ”kapa za tuširanje” i to sa sigurnošću od 89.84%. Druga klasa za koju je najsigurnija je klasa ”perika” i to sa sigurnošću od 4.52%. Preko lokalizacijske mape, odnosno *Grad-CAM*-a se jasno vidi kako se mreža fokusira na nebitne dijelove slike. Izvor originalne fotografije: [11].



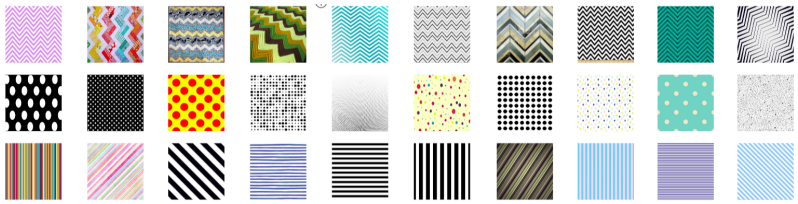


Slika 4.5: Slika koja ne predstavlja ništa i njena pripadna lokalizacijska mapa *GoogleNet* konvolucijske neuronske mreže koja je napravila predikciju ove slike. *GoogleNet* je u ovom slučaju klasificirao ovu sliku kao ”labirint” i to sa sigurnošću od 80.15% te kao ”moždani koralj” sa sigurnošću od 13.86%. s desne strane vidimo lokalizacijsku mapu koja nam ne otkriva razumna objašnjenja. Izvor originalne fotografije: [5].

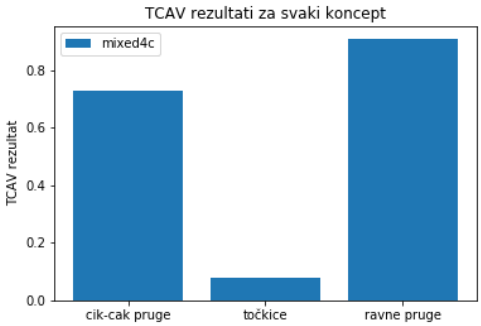
metoda ([17]). Napominjemo da je kod prilagođen autorovim potrebama odnosno da je kod mijenjan kako bi se dobili rezultati opisani u ovom radu. U ovom primjeru se koristimo bibliotekom *TensorFlow* koja je uz *PyTorch* jedna od najpoznatijih biblioteka namijenjena dubokom učenju.

Proveli smo *TCAV* metodu i dobili očekivane rezultate, Naime, kako se može i vidjeti na slici 4.7, koncept ”ravne pruge” najviše utječe na rezultat predikcije, ali i koncept ”cik-cak pruge” igra bitnu ulogu u tome. Koncept ”točkice” vrlo malo sudjeluje u predikciji.

Ovi eksperimenti su vrlo korisni jer ljudi ne razumiju objekte oko sebe preko brojeva (odnosno težina u neuronskoj mreži) nego većinu stvari shvaćaju kroz pojam koncepta. Upravo zato je ova metoda vrlo popularna u današnje vrijeme jer je vrlo bliska i korisniku koji toliko i ne razumije strojno učenje općenito.



Slika 4.6: Slike koje definiraju tri vrste koncepata korištenih u eksperimentu. U prvom redu se nalaze slike koje definiraju koncept ”cik-cak pruge”. U drugom redu se nalaze slike koje definiraju koncept ”točkice” dok je posljednjim redom definiran koncept ”ravne pruge”. Slike su pronađene tražilicom *Google*.



Slika 4.7: Prikaz rezultata *TCAV* metode. Na dijagramu se vidi kako koncept ”ravne pruge” vrlo utječe na predikciju klase ”zebra”, a i koncept ”cik-cak pruge” igra veliku ulogu u tome. Za koncept ”točkice” možemo reći kako vrlo malo utječe na rezultat predikcije.

Poglavlje 5

Zaključak

U ovom radu smo predstavili brojne metode interpretabilnosti konvolucijskih neuronskih mreža što je još uvijek relativno novo područje istraživanja. Vizualizacija CNN reprezentacije predstavlja prva istraživanja tog područja. Tu se najviše ističu metode koje koriste *saliency* mape te se te metode u manjoj mjeri koriste danas jer postoje bolje metode koje rješavaju neke mane prisutne u spomenutim metodama. Dijagnoza CNN reprezentacije nam je predstavila dvije metode koje su popularne i danas te koje smo najviše analizirali u ovome radu. Radi se o *Grad-CAM* i *TCAV* metodama koje su se na neki način približile i korisnicima koji nemaju puno znanja o strojnom učenju. Posebno je zanimljiva metoda *TCAV* koja pokazuje utjecaj nekog koncepta na predikciju CNN modela. Metoda koja "otpetljava" CNN reprezentaciju u eksplanatoran graf je napravila velik pomak na području interpretabilnosti jer nam daje značajne informacije o značajkama koji se pojavljuju u CNN modelu poput njihovog tipa, prostorne povezanosti ili simultanog pojavljivanja s drugim značajkama. Metoda učenja neuronske mreže koja ima interpretabilnu reprezentaciju ima dosta potencijala jer već kod treniranja pokušava dobiti interpretabilnu CNN reprezentaciju. Interpretabilno *middle-to-end* učenje se smatra budućom najpopularnijom metodom među znanstvenicima na području strojnog učenja. Razlog tome je što znanstvenik može "usmjeriti" učenje u smjeru u kojem želi te ova metoda zahtijeva vrlo malo dodatnih anotacija. Sve ove metode nam daju dodatne informacije o CNN modelu, ali i općenito o modelu strojnog učenja, te smatramo da bi one trebale biti sastavni dio razvoja istih.

Bibliografija

[1] Agrawal A, Lu J et al., *VQA: Visual Question Answering*, (2015), https://arxiv.org/abs/1505.00468.

[2] Dosovitskiy A i Brox T, *Inverting visual representations with convolutional networks*, (2016), https://arxiv.org/abs/1506.02753.

[3] Krizhevsky A, Sutskever I, E. Hinton G et al., *ImageNet Classification with Deep Convolutional Neural Networks*, (2012), https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[4] Mahendran A i Vedaldi A, *Understanding deep image representations by inverting them*, (2015), https://doi.org/10.1109/CVPR.2015.7299155.

[5] Nguyen A, Yosinski J i Clune J, *Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images*, (2014), https://arxiv.org/abs/1412.1897.

[6] Kim B, Wattenberg M et al., *Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)*, (2017), https://arxiv.org/abs/1711.11279.

[7] Zhou B, Khosla A et al., *Learning Deep Features for Discriminative Localization*, (2015), https://arxiv.org/abs/1512.04150.

[8] Szegedy C, Liu W et al., *Going Deeper with Convolutions*, (2014), https://arxiv.org/abs/1409.4842.

[9] Szegedy C, Zaremba W, Sutskever I et al., *Intriguing properties of neural networks*, (2014), http://arxiv.org/abs/1312.6199.

[10] Silver D, Huang A et al., *Mastering the game of Go with deep neural networks and tree search*, (2016), https://www.ncbi.nlm.nih.gov/pubmed/26819042.

[11] Julia Evans, *How to trick a neural network into thinking a panda is a vulture*, 2018, https://codewords.recurse.com/issues/five/why-do-neural-networks-think-a-panda-is-a-vulture, posjećena 2019-10-31.

[12] Iandola F, Han S et al., *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and less than 0.5MB model size*, (2016), https://arxiv.org/abs/1602.07360.

[13] Huang G, Liu Z et al., *Densely Connected Convolutional Networks*, (2016), https://arxiv.org/abs/1608.06993.

[14] Lakkaraju H, Kamar E, Caruana R et al., *Identifying unknown unknowns in the open world: representations and policies for guided exploration*, (2017), https://arxiv.org/abs/1610.09064.

[15] Misra I, Zitnick CL et al., *Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels*, (2015), https://arxiv.org/abs/1512.06974.

[16] Su J, Vargas DV i Kouichi S, *One pixel attack for fooling deep neural networks*, (2017), http://arxiv.org/abs/1710.08864.

[17] Wexler J, Kim B et al., *Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV) [ICML 2018]*, 2018, https://github.com/tensorflow/tcav, posjećena 2019-10-31.

[18] Springenberg JT, Dosovitskiy A, Brox T et al., *Striving for simplicity: the all convolutional net*, (2015), https://arxiv.org/abs/1412.6806.

[19] He K, Zhang X et al., *Deep Residual Learning for Image Recognition*, (2015), https://arxiv.org/abs/1512.03385.

[20] Simonyan K, Vedaldi A i Zisserman A, *Deep inside convolutional networks: visualising image classification models and saliency maps*, (2013), http://arxiv.org/abs/1312.6034.

[21] Simonyan K i Zisserman A, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, (2014), https://arxiv.org/abs/1409.1556.

[22] WonKwang Lee, *A Simple pytorch implementation of GradCAM and GradCAM++*, 2018, https://github.com/1Konny/gradcam_plus_plus-pytorch, posjećena 2019-10-23.

[23] Zintgraf LM, Cohen TS et al., *Visualizing Deep Neural Network Decisions: Prediction Difference Analysis*, (2017), https://arxiv.org/abs/1702.04595.

[24] Aubry M i Russell BC, *Understanding deep features with computer-generated imagery*, (2015), https://doi.org/10.1109/ICCV.2015.329.

[25] Zeiler MD i Fergus R, *Visualizing and understanding convolutional networks. European Conf on Computer Vision*, (2014), https://doi.org/10.1007/978-3-319-10590-1_53.

[26] Ribeiro MT, Singh S i Guestrin C, *"Why should I trust you?" explaining the predictions of any classifier*, (2016), https://doi.org/10.1145/2939672.2939778.

[27] Koh P i Liang P, *Understanding black-box predictions via influence functions*, (2017), https://arxiv.org/abs/1703.04730.

[28] Kindermans PJ, Schütt KT et al., *Learning how to explain neural networks: PatternNet and PatternAttribution*, (2017), https://arxiv.org/abs/1705.05598.

[29] Zhang Q, Cao R i Shi F, *Interpreting CNN knowledge via an explanatory graph*, (2018b), https://arxiv.org/abs/1708.01785.

[30] Zhang Q, Cao R, Wu YN et al., *Growing Interpretable Part Graphs on ConvNets via Multi-Shot Learning*, (2017), https://arxiv.org/abs/1611.04246.

[31] Zhang Q i Zhu S, *Visual Interpretability for Deep Learning: a Survey*, (2018), https://arxiv.org/abs/1802.00614.

[32] Zhang Q, Wang W i Zhu SC, *Examining CNN representations with respect to dataset bias*, (2018a), https://arxiv.org/abs/1710.10577.

[33] Zhang Q, Wu YN i Zhu SC, *Interpretable convolutional neural networks*, (2018), https://arxiv.org/abs/1710.00935.

[34] Fong RC i Vedaldi A, *Interpretable explanations of black boxes by meaningful perturbation*, (2017), https://doi.org/10.1109/ICCV.2017.371.

[35] Selvaraju RR, Cogswell M, Das A et al., *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization*, (2016), https://arxiv.org/abs/1610.02391.

[36] Bolukbasi T, Chang KW et al., *Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings*, (2016), https://arxiv.org/abs/1607.06520.

[37] Muneeb ul Hassan, *VGG16 – Convolutional Network for Classification and Detection*, 2018, https://neurohive.io/en/popular-networks/vgg16/, posjećena 2019-10-23.

[38] Yulong Wang, *Pytorch-Visual-Attribution*, 2018, https://github.com/yulongwang12/visual-attribution, posjećena 2019-10-23.

[39] LeCun Y, Bottou L et al., *Gradient Based Learning Applied to Document Recognition*, (1998), https://ieeexplore.ieee.org/document/726791.

[40] Lu Y, *Unsupervised learning on neural network outputs*, (2015), http://arxiv.org/abs/1506.00990.

Sažetak

Duboke neuronske mreže postižu zavidne performanse u područjima poput klasifikacije i predikcije. U današnje vrijeme čak i dostižu sposobnosti samih ljudskih eksperata. No, još uvijek se u najčešćem broju primjena ti modeli smatraju crnom kutijom za koje često ni sam autor ne može dovoljno dobro objasniti kako i zašto funkcioniraju. Upravo zbog toga se javlja potreba za tehnikama koje će na ljudski razumljiv način interpretirati modele, odnosno omogućiti ljudsku/ekspertsku procjenu njihove robusnosti. U ovom radu predstavljamo takve metode interpretabilnosti konvolucijskih neuronskih mreža što je još uvijek relativno novo područje istraživanja. Najviše proučavamo metode *Grad-CAM* i *TCAV* za koje provodimo i pripadne eksperimente. Osim njih spominjemo i metode vizualizacije CNN reprezentacije (ponajviše *saliency* mape), "otpetljavanje" CNN reprezentacije u eksplanatorne grafove, učenje neuronske mreže koja ima interpretabilnu reprezentaciju te interpretabilnost *middle-to-end* učenja.

Summary

Deep neural networks achieve enviable performance in terms of classification and prediction. Nowadays we are even able to find those which are more efficient than human experts in certain fields. But, in a large number of applications, those models are considered as ”black box” since neither their author can’t explain them properly. Because of that, there is a need for techniques which can interpret models in a way understandable for humans and scientists as well. In this work, we present those methods of convolutional neural network interpretability which is still a young field of research. Most of all, we study *Grad-CAM* and *TCAV* methods for which we conduct experiments. Besides them, we study methods of CNN representation visualization (mostly *saliency maps*), ”disentangling” CNN representation into explanatory graphs, learning neural networks with interpretable representations and network interpretability for *middle-to-end* learning.

Životopis

Rođen sam 20. kolovoza 1995. godine u Osijeku. Osnovnu školu sam pohađao u Čepinu nakon koje sam upisao III. gimnaziju u Osijeku. Uvijek sam uživao u matematici te sam već od osnovne škole svake godine išao na matematička natjecanja. Nakon srednje škole upisujem se na studij matematike na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta. Nakon preddiplomskog studija, odlučujem se za diplomski studij Računarstva i matematike na istom odsjeku.

Na trećoj godini se zapošljavam u tvrtci MicroBlink gdje radim skoro dvije godine. U tom periodu počinjem shvaćati kako se želim baviti strojnim učenjem u svojoj karijeri te se počinjem sve više angažirati na tom području u svoje slobodno vrijeme. Nakon obrane diplomskog rada, počinjem raditi u tvrtci Visage Technologies u diviziji za autonomnu vožnju.