# Robust ML Challenge
# Andrew-Descents


Ante Buterin
Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu
Zadar, Hrvatska
Email: antebuterintubi@homtail.com

Florijan Iljazović
Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu
Zagreb, Hrvatska
Email: florijan.iljazovic@gmail.com

Filip Skrinjar
Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu
Rijeka, Hrvatska
Email: skrinjarf@gmail.com



***Sažetak*—Prezentiramo rješenje zadatka s natjecanja Mozgalo pod nazivom "Robust ML Challenge". Problem zadatka je klasifikacija slika računa, gdje preko obilježja računa (prvenstveno mislimo na logo trgovačkog lanca) određujemo kojem trgovačkom lancu pripada. Naše rješenje je end-to-end rješenje, koje koristi ResNet duboku konvolucijsku arhitekturu. Postignuta je velika preciznost, a zadovoljeni su i glavni uvjeti natjecanja: robusnost i efikasnost.**


## I. Uvod

Strojno učenje je u posljednjim godinama u području računalnog vida napravilo svojevrsnu revoluciju. Na nekim javno dostupnim skupovima podataka postižu se rezultati bolji od čovjeka što ukazuje na velik potencijal za ozbiljnu industrijsku primjenu. Jedna od tih primjena je i dubinska analiza tržišta, gdje se prikupljanjem podataka direktno od korisnika mogu donositi zaključci i predviđanja.

Jedan od ovogodišnjih zadataka na Mozgalu, u organizaciji tvrtke Microblink a pod nazivom "Robust ML Challenge", svoju motivaciju nalazi upravo u tome. Nakon kupovine u nekom trgovačkom lancu, korisnici fotografiraju svoj račun, i pošalju ga na skeniranje. Sastavna komponenta analize računa je i klasifikacija računa ovisno o tome kojem trgovačkom lancu pripadaju. Upravo to je bio zadatak natjecanja.

Mi smo se odlučili za ovaj problem prvenstveno jer smo smatrali da je to dobra prilika za upoznavanje s područjem računalnog vida, te njegovim aktualnim temama i metodama. Posebno, ciljevi ovog rada su upoznavanje sa aktualnim arhitekturama konvolucijskih neuralnih mreža, metodama transfer learninga, ali i izrada ML pipelinea dovoljno dobrog za sudjelovanje na nekom natjecanju.

## II. Opis Problema

Zadatak je u teoriji jednostavan - klasifikacija slika računa na temelju toga kojem trgovačkom centru pripdaju. Osnovno obilježje računa je logo trgovačkog centra. Skup za trening je označen i sastoji se od 45 000 slika računa iz 25 različitih klasa (različitih lanaca). Skup za testiranje sastoji se od još 20 000 slika računa, od kojih je jedna polovica bila dostupna natjecateljima za evaluiranje svojih rješenja, a druga je služila za ocjenjivanje rješenja.

Slika 1. Primjer računa iz testnog skupa


Problem se komplicira kada pogledamo same slike računa, kao što vidimo na primjeru slike iznad. One nisu nastale u idealnoj okolini, već su to stvarne slike koje su ljudi slikali mobilnim uređajima. Iz tog razloga računi na slikama su zarotirani, translatirani, deformirani, zguzvani, sakriveni, prekriveni itd. Pozadina i osvjetljenje na slikama varira, a na nekim slikama ima i više od jednog računa. Zatim, polovica slika iz testnog skupa ne pripada niti jednoj od 25 zadanih klasa, iako su to i dalje računi. Njih je potrebo klasificirati u 26. klasu, koju zovemo **Other**.

To sugerira da je jedan od ključnih zahtjeva zadatka robusnost. Potrebno je izraditi rješenje koje će se dobro ponašati u nepredviđenim (a nekada i nepredvidivim) okolnostima. Zatim, rješenje mora biti efikasno, odnosno klasifikator mora biti brz.

## III. Pristup problemu

Od samog početka smo znali da ćemo za klasifikaciju koristiti duboke konvolucijske arhitekture. Vidjeli smo rezultate ImageNet-ovog godišnjeg natjecanja, gdje se mreže mnogih velikih kompanija natječu u klasifikaciji slika. 2017. godine

29 od 38 timova je imalo ispod 5% pogreške, što u nekim slučajevima znači da su uspješnije od ljudi. Poslijednje dvije godine pobjednik je bio Microsoftov ResNet. ResNet arhitektura koja je pobijedila je konvolucijska neuralna mreža sa 152 sloja. Naravno za naš problem to je prevelika mreža, koju bismo lako prenaučili. Stoga smo odlučili da ćemo testirati najmanju varijantu ResNet arhitekture, **ResNet18**, koja broji 18 slojeva.

Također, vođeni onime što smo čuli od organizatora, smatrali smo da će biti potrebno na neki način lokalizirati logo trgovačkog lanca prije klasifikacije, tj. identificirati dio računa na kojem se logo nalazi, kako bismo pospješili klasifikaciju nezgodnih računa. Napomenimo da je eksplicitno zabranjen bio sljedeći pristup: ručno označiti neke od računa na način da istaknemo logo, pa onda trenirati mrežu da nauči prepoznavati gdje se taj logo nalazi. Organizatori su ponudili nekoliko ideja, prije svega izdvajanje računa iz pozadine metodama tresholdinga (pretpostavka je da je račun svjetliji od pozadine, i to se koristi za izdvajanje), te korištenje autokorelacije (uspoređivanje slike računa sa slikom nekog logoa i traženje preklapanja). Nismo bili zadovoljni tim metodama - pretpostavka o svjetlini u velikom broju slučajeva nije bila ispunjena, a postupak "autokoreliranja" loše reagira na različitu skaliranost. Također, vrlo brzo smo došli do sljedećeg zaključka - klasifikacija ne mora nužno biti bazirana samo na logu. Naime, različiti trgovački lanci koriste različite obrasce za izradu svojih računa, pa je moguće da određeni font ili struktura računa također karakterizira klasu. Stoga smo se odlučili za pokušaj preskakanja procesa lokalizacije, tj odlučili smo trenirati mrežu na neobrađenim računima i vidjeti kako će se ponašati. Odmah smo postigli jako dobre rezultate, pa smo prihvatili taj pristup i u daljnjem.

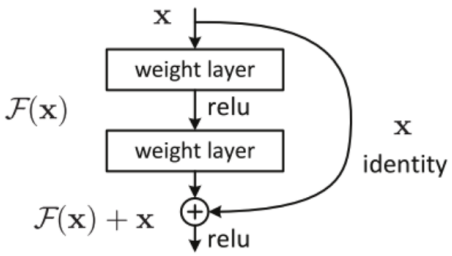
## IV. ResNet

Prema univerzalnom aproksimacijskom teoremu, uz dovoljno kapaciteta, znamo da feedforward mreža sa jednim slojem može naučiti reprezentirati bilo koju funkciju. No taj sloj može biti prevelik i mreža je tada sklona overfitu. Zbog toga, danas je trend da se koriste duboke neuralne mreže. Od AlexNet arhitekture, state-of-the-art konvolucijske neuralne mreže (CNN) pokušavaju biti što dublje. Dok je AlexNet imao samo 5 konvolucijskih slojeva, VGG i GoogleNet su imali 19 i 22 sloja respektivno.

Međutim, puko nizanje slojeva nije bezopasan proces, pa povećanje dubine mreže nije jednostavan zadatak. Duboke mreže su teške za učenje zbog problema nestajućeg gradijenta. U procesu backpropagacije, kako se gradijent prenosi od dubljih prema plićim slojevima, uzastopno množenje može dovesti do toga da gradijent postane toliko malen da više nema utjecaja na učenje plićih slojeva. Stoga, što je mreža dublja, njena performansa počinje s vremenom stagnirati ili čak i opadati. Glavna ideja ResNet arhitekture je korištenje tzv. "identity shortcut connection", koji preskaču jedan ili više slojeva, kao što je pokazano na slici (2).

Dajemo sada formalniji zapis prethodne ideje. Promotrimo forward propagaciju od sloja $l$ do sloja $l+2$. Neka je $W^{[l+1]}$ matrica težina između sloja $l$ i $l+1$, $a^{[l+1]}$ je aktivacija koja izlazi iz sloja $l+1$, $b^{[l+1]}$ je *bias* težina na sloju $l+1$ te je $g$ aktivacijska funkcija (u našem slučaju RELU aktivacija). Stadnardne mreže bez "preskoka" daju sljedeće aktivacije:

Slika 2. rezidualni blok




$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]}$$
$$a^{[l+1]} = g(z^{[l+1]}) \tag{1}$$

$$z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]}$$
$$a^{[l+2]} = g(z^{[l+2]}) \tag{2}$$

Međutim ResNet ima suptilnu modifikaciju pri računanju posljednje aktivacije u (2):

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]}) \tag{3}$$

Kreatori ResNeta tvrde da dodavanje slojeva nebi smjelo oslabiti preformanse mreže, iz razloga što bi mogli jednostavno dodavati mapiranje identitetom (slojevi koji ne rade ništa) na trenutnu mrežu, i rezultirajuća mreža bi imala iste performanse.

Promotrimo intuiciju iza tih tvrdnji. Kao što smo rekli, duboke neuronske mreže pate od problema nestajućeg gradijenta. To znači da će nam težine mreže opadati jako blizu nuli. To najčešće dovodi do loših performansi mreže jer tako ne mogu ništa "novo" naučiti. Pogledajmo jednostavnom matematičkom intucijom zašto ResNet taj problem rješava. Uvrstimo jako male težine u (3):

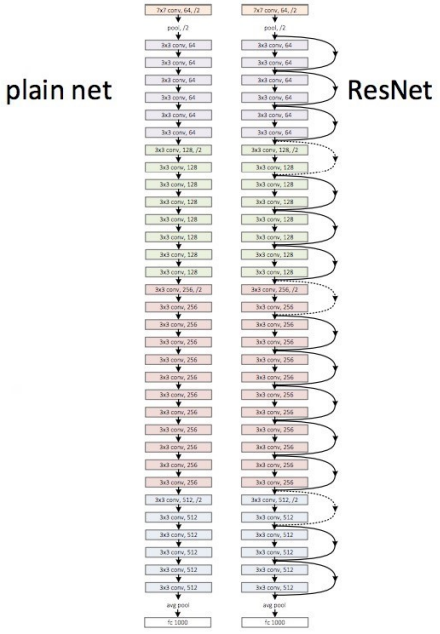
$$\begin{aligned} a^{[l+2]} &= g(z^{[l+2]} + a^{[l]}) = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + +a^{[l]}) \\ &= \left\{W^{[l+2]} = 0, \quad b^{[l+2]} = 0\right\} = g(a^{[l]}) = a^{[l]} \end{aligned} \tag{4}$$

Vidimo da će sada naše aktivacije biti barem jednake onima od prije 2 sloja čime će mreža u dubljim slojevima moći naučiti najmanje identitetu, za razliku od ekvivalentne mreže bez preskoka.

## V. Treniranje i prvi rezultati

Za implementaciju smo koristili PyTorch paket u Pythonu 3, jer nam on daje high level API uz brzinu ekvivalentnu TensorFlow biblioteci. Za samo učenje koristili smo transfer learning, na način da smo učitali istrenirani ResNet18 model te smo modificirali zadnji sloj moreže da ima 25 izlaza, po jedan za svaku klasu. Potom smo "odledili" sve težine tako da se

Slika 3. usporedba arhitektura bez i s preskočenim konekcijama




Slika 4. Primjer slika nakon augmentacije

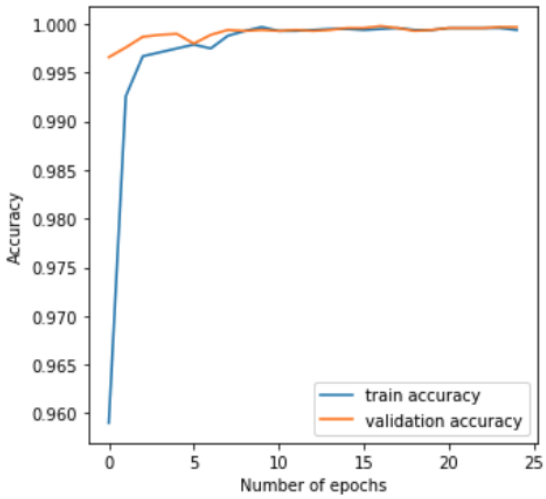

za vrijeme treniranja u fazi *backpropagacije* gradijent prenosi kroz cijelu mrežu a ne samo kroz klasifikator. Odabrali smo takav pristup jer je kombinacija onog što želimo - sačuvati korisne stvari naučene na ImageNet-ovim slikama (elementarni oblici, rubovi) i naučiti mrežu za specifičan problem koji imamo (računi). Naime, prvi slojevi mijenjat će se malo, a kasniji slojevi koji služe tumačenju prethodno prepoznatih značajki se dosta mijenjaju. Kao i sa svakom dubokom neuralnom mrežom, potencijalni problem je overfit. Kako bi što više ublažili taj problem, radimo augmentaciju skupa za treniranje. Transformacije koje smo radili nad slikama su:

- Resize na 224x224, što je minimalni dopušteni ulaz u ResNet mrežu, prema službenoj dokumentaciji.
- Proizvoljna rotacija za kut od -30 stupnjeva do +30 stupnjeva.
- ColorJitter(), funkcija koja radi nasumično mjenjanje kontrasta slike, oštrine, dodavanje boja ...
- Normalizacija slike na ResNet-ov standard.

Na slici (4) možemo vidjeti primjer kako izgledaju slike nakon nakon transformacija.

Navedene transformacije rađene su nad svakom slikom prilikom učitavanja slike, tako da kod svakog ponovnog učitavanja iste slike se, zbog proizvoljnosti pojedinih transformacija, primjenjuju drugačije transformacije. Time jako ublažavamo

Slika 5. Točnost po epohama




prenaučenost, jer mreža sada ne zna točne lokacije gdje se javlja neki feature kao npr. logo, jer uvijek imamo neku rotaciju.

Dobivenih 45000 slika smo podjelili na 35000 slika za treniranje i 10000 slika za validaciju. Učitavali smo podatake i provodili trening koristeći pogodne alate dostupne u okviru paketa **PyTorch**, kao što je DataLoader(). Konkretno, trening se zasniva na *batchevima* od 4 slike, a za svako epohu uzimamo nasumično 35000 slika za treniranje i 10000 slika za validaciju iz odgovarajućih skupova. Za *loss*-funkciju smo odabrali **cross-entropy loss**. Ona je osmišljena kako bi učenje bilo što brže jer ne uzima u obzir derivaciju sigmoidne funkcije. Za kontroliranje stope učenja koristili smo eksponencijalni *learning rate scheduler*, koji eksponencijalno smanjuje stopu učenja nakon odgovarajućeg broja epoha. Optimizacijska funkcija koju smo koristili je stohastički gradijentni spust (SGD) (već smo naveli da je veličina *batcha* 4), kako je za ovakvu količinu podataka on puno pogodniji od običnog gradijentnog spusta. Sav izračun se izvršava na grafičkoj procesorskoj jedinici (specifikacija u poglavlju (VIII) )

Na slici 5 prikazana je krivulja učenja. Vidimo da već nakon jedne epohe postižemo veliku točnost, a nakon par epoha točnost na skupu za validaciju prelazi preko 99.9%. Uočavamo da nikako nismo prenaučili mrežu, jer je preciznost velika i na skupu za trening i na skupu za validaciju, prakticki bez razlike.

Iz rezultata zaključujemo da je ResNet18 doista i više nego dovoljan za vrhunsko raspoznavanje treniranih klasa. Međutim, u obzir moramo uzeti i Other klasu.

## VI. Detekcija klase Other i rezultati

Kako detektirati da dana slika ne pripada niti jednoj od treniranih klasa? Izlaz naše mreže je vektor $x \in \mathbb{R}^{25}$. Klasificiramo tako da odredimo argument maksimuma vektora $x$. Prva ideja za detekciju other klasa je da odaberemo nekakvu granicu (treshold) sigurnosti mreže $T$, pa klasificiramo u Other ako je

vrijednost maksimuma manja od tresholda, tj ako $\max x < T$. Inače klasificiramo normalno.

Treshold smo odabrali iz vjerojatnosne interpretacije outputa. Naime, kada za zadnji sloj stavimo softmax, tada elemente od $x$ kao vjerojatnosti. Sada ako želimo "sigurnost" od 90%, stavimo $T = 0.9$. Upravo to smo i napravili, pa testirali. Rezultati nisu bili zadovoljavajući, F1 norma na testnom skupu ispala je oko 0.6.

Međutim, daljnjom inspekcijom izlaza neuralne mreže za različite slike uočili smo da neke klase prirodno poprimaju dosta manje vrijednosti u svojim komponentama od nekih drugih, tj. uočili smo da neke klase standardno češće (neopravdano) "lete" u Othere nego neke druge. Prirodno se nametnuo idući korak - za svaku klasu definirat ćemo njen treshod. Dakle klasifikacija se odvija na sljedeći način: potražimo maksimum, i onda maksimum usporedimo sa odgovarajućim tresholdom (ovisno o tome koji je argument maksimuma, odnosno koja je klasa).

Kako određujemo te tresholde ? Početne, grube procjene odredili smo statistički. Na validacijskom skupu promatrali smo distribucije vektora izlaza za pojedine klase, točnije promatrali smo distribcije maksimalnih elemenata pojedinih klasa. Tako smo spuštanjem za par standardnih devijacija od aritmetičkih sredina dobili osnovne procjene tresholda.

Međutim, bila je potrebna direktna intervencija u tresholde. Na osnovi uzorka od oko 50 do 100 slika iz testnog skupa, revidirali samo tresholde po potrebi. Na primjer, za klasu JewelOsco uočili smo da je potreban izrazito visok treshold. Većina ih se kretala oko 10. (napominjemo da u ovom pristupu nismo radili završnu softmax transformaciju). Pitali smo se kako će ova metoda generalizirati na ostatku testnog skupa, a onda i na skupu za ocjenjivanje. Pokazalo se da metoda radi dobro, i postignut je F1 rezultat od 0.89. Identican rezultat postignut je i na skupu za ocjenjivanje. Taj rezultat bio je dovoljan za plasman u finale natjecanja Mozgalo.

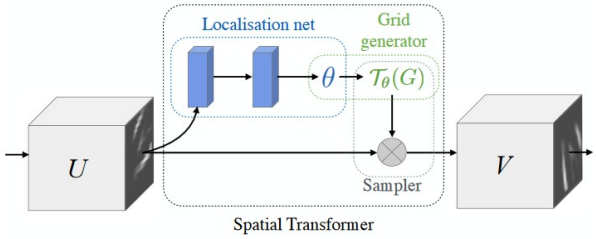
## VII. Neuspjeli pokušaj - Spatial transformer network

Jedna od bitnih značajki kvalitetnog modela u području računalog vida je prostorna invarijantnost. Duboke konvolucijske arhitekture standardno nailaze na poteškoće u tom segmentu. Neke od klasičnih metoda za adresiranje tog problema su:

- Max pooling layeri - dodavanje takvih slojeva znači da se gubi potpuno točna informacija o poziciji, a u korist prostorne invarijantnosti.
- Augmentacija skupa za treniranje - proširivanje skupa za trening potencijalno teškim primjerima.

Max pooling slojevi nisu idealno rješenje. Po konstrukciji uništavaju informaciju, a prostorna invarijantnost se postiže tek u dubljim slojevima mreže. Naime, da bismo utjecali na prve slojeve, veličina prozora iz kojeg čuvamo maksimum mora biti poprilično velika, što vodi još većem gubitku informacije. Augmentacije skupa za treniranje bismo se također htjeli riješiti ako je moguće, naprosto s ciljem skraćivanja vremena treniranja. Sada ćemo ukratko prezentirati jedno od boljih

Slika 6. Spatial transformer




alternativnih rješenja. **Spatial transformer** (ST) je modul posebno dizajniran tako da se može umetnuti na proizvoljno mjesto duboke konvolucijske arhitekture, i pružiti prostornu invarijantnost modelu. Mreže koje sadržavaju taj modul nazivaju se **Spatial Transformer Networks** (STN). Na donjoj slici je shematski prikaz građe tog modula. Opišimo djelovanje u nekoliko koraka:

- Ulaz - slika, odnosno općenitije gledano, neka mapa značajki (ST se ne mora nalaziti na početku).
- Mapa se daje kao input **lokalizacijskoj mreži**. To može biti kovolucijska, ili neka druga mreža. Njezin output su parametri 2D transformacije koju je potrebno primijeniti. Konkretno, output čini 6 brojeva koji, posloženi u $2 \times 3$ matricu (označimo ju sa $\theta$) predstavljaju matricu transformacije koju treba primijeniti.
- Primjenjuju se dobivene 2D transformacije, na sljedeći način. Slici se pridružuje 2D koordinatna mreža, gdje svaki čvor $x \in \mathbb{R}^2$ odgovara jednom pikselu. Nova pozicija čvora je $\theta x$. Sada su tako transformirani čvorovi upravo pikseli koje želimo prikazati. Zadnji korak se odvija kroz tzv. sampler, u njemu se zapravo odvija interpolacija iz vrijednosti čvorova u cjelobrojne vrijednosti piksela, i na izlazu cijelog modula je transformirana mapa značajki.

Ideja je da se na izlazu ST modula nalazi modficirana slika, i to modificirana na način da se pospješuje daljnja klasifikacija. U našem slučaju, nadamo se da će se na izlazu ST-a naći slika u kojoj je fokusiran i poravnat logo računa. Važna karakteristika STN-a je što su transformacije **naučene** (preko lokalizacijske mreže), kao i sve ostale težine u arhitekturi.
Implementirali smo ST modul i pokušali ga uključiti u model, neposredno prije ResNeta. Nismo dobili zadovoljavajuće rezultate. Vodeća pretpostavka o tome gdje smo pogriješili je ta da smo istodobno trenirali ST + ResNet. Naknadnim čitanjem originalnog članka ([6]) zaključili smo da je potrebno prvo trenirati klasifikator, pa potom ST i klasifikator zajedno. Naša interpretacija je ova: kod istodobnog učenja na početku ResNet nezna klasificirati račune jer ima težine koje odgovaraju klasifikaciji slika sa ImageNet-a, a isto tako niti ST modul nezna odrediti gdje se nalazi logo. Pa onda u slučaju krivog određivanja klase ne znamo je li krivac ResNet ili ST modul.

## VIII. Mjerenja

Kao što je navedeno u poglavlju (V), treniranje i evaluacija rješenja je rađena na grafičko procesorskoj jedinici (GPU),

no po završetku projekta testirali smo kako naše rješenje radi kada koristimo procesor (CPU), kako bi dobili bolji uvid u zahtjevnost programa. Mjerenja iz tablice (7) dobivena su na validacijskom skupu od 10000 slika. Dakle vidimo da i na relativno jeftinim računalima naš model može klasificirati i više od 2 slike po sekundi. To znači da je ovakvu mrežu moguće koristiti i za klasifikaciju uživo.

Slika 7. mjerenja na raznim hardwareima

| | Nvidia 1060 | Nvidia 960 | Intel i5-6500 |
|---|---|---|---|
| Memorija | SSD | HDD | HDD |
| Vrijeme | 7m 23s | 25m 13s | 2h 30m |
| Točnost | 99.7% | 99.7% | 99.5% |

## IX. Zaključak i moguća poboljšanja

U konačnici, možemo reći da smo odabirom pravog modela, odnosno prave arhitekture, u potpunosti zaobišli problem lokalizacije loga. Također, ispravno smo trenirali model, i nismo ga prenaučili. Postigli smo vrhunske rezultate na zadanih 25 klasa, kako na validacijskom skupu tako i na skupovima za testiranje. Čitav model je ostao veoma brz i lagan zbog korištenja manje varijante ResNet arhitekture.

Jedini problem ovog rješenja je detekcija Other klase. Naime, neke jako dobre pogotke našeg modela "gubimo" tako što ostanu "tresholdani" i završe u Other klasi. Uočimo još jedno ograničenje ovakve detekcije Othera.


Slika 8. Račun iz treniranih klasa

Na slici 8 nalazi se jedan od računa iz treniranih klasa, a na slici 9 je račun iz Other klase. Uočavamo da oba računa imaju prakticki identičan logo, ali dolaze iz različitih firmi. Ne postoji razuman treshold koji će ovdje dati dobar rezultat. Alternativne metode za detekciju Othera:

- Za danu sliku napraviti više forward-passova (moguće jer se forward pass odvija brzinom i do 7 slika u sekundi (na GPU)), pa uzeti najčešći rezultat kao konačni. Ovaj pristup je u fazi testiranja, zasad se čini da daje bolje rezultate.
- Izlaz mreže shvatiti kao element 25-dimenzionalnog prostora, te na temelju udaljenosti od vektora iz skupa za trening/validaciju donijeti zaključak. Naime, gledanjem samo maksimalne komponente gubimo moguće korelacije među klasama.


Slika 9. Račun iz klase Other

Prikažimo sada i jedan pogodak naše mreže na temelju kojeg ćemo djelomično demonstrirati robusnost modela.


Slika 10. Račun iz klase Other

Iako logo Walmart-a ulazi u kadar, naš model ispravno klasificira ovaj primjer kao Other.

## Literatura


[1] Službene radionice sa natjecanja Mozgalo
[2] *Machine Learning*, Coursera, Andrew Ng
[3] *Deep residual learning for image recognition*, Microsoft research, 2015
[4] službeni tutoriali za PyTorch $https://pytorch.org/tutorials/$
[5] *Spatial Transformer Networks*, Google Deep Mind, 2016