|  |  |  |
| --- | --- | --- |
| pmf | УНИВЕРЗИТЕТ У НОВОМ САДУ  ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ  ДЕПАРТМАН ЗА МАТЕМАТИКУ И ИНФОРМАТИКУ | grbUNS |

**Konvolutivne mreže**

Projekat iz predmeta Neuronske mreže

**Ime i prezime: Marko Popović**

Sadržaj

1. **Uvod**3

2. **Deo 1**6

**2.1 Dense** 6

**2.2 Dense\_Convnet**7

3 **Deo 2**9

**3.1 Constants** 9

**3.2 Buiding\_model**10

**3.3 Create\_dataset\_folder**11

**3.4 Network\_1**13

**3.5 Augmentation\_example**15

**3.6 Network\_2**17

**3.7 Vizualizing**19

**3.8 Vizualizing\_heatmap**21

**3.9 Vizualizing\_tabelarno** 23

**1. Uvod**

|  |
| --- |
| [http://pc.pcpress.rs/arhiva/images/broj/175/100_slika01.jpg](http://pc.pcpress.rs/arhiva/images/broj/175/100_slika01-big.jpg) |
|  |

Kao kod svake nove tehnologije, i ovde je potreban napor da bi se shvatio pricip i filozofija rada. Neuronska mreža je napravljena po uzoru na biološko nervno tkivo i u mnogo čemu ga podražava.

Svako od nas u mozgu ima oko sto milijardi neurona, a na svaki od tih neurona dovodi se oko 10.000 signala sa drugih neurona. Sve to čini veoma složenu mrežu, iz koje stižu naše refleksne radnje, instinkti, inteligencija, svest, znanje, sećanje na razne događaje i iskustva, pa i naše celokupno psihičko biće.

Šematski prikaz veštačkog neurona (na slici) je praćen matematičkim opisom koji ne obećava baš lako razumevanje nekome kome nije blizak jezik matematike. Zato ćemo ga prevesti na „naš“ jezik: neuron ima više ulaza i jedan izlaz. Svaki od ulaza (nacrtana su četiri, a može da ih bude manje ili mnogo više) dovodi signal sa nekog od prethodnih neurona, i taj signal se množi sa tzv. težinskim koeficijentom (ovaj izraz zapamtite, jer ćemo se vraćati na njega).

Svi ulazi (a za svaki od njih postoji zaseban težinski koeficijent) se sabiraju, što nam govori simbol ∑ (suma) nacrtan u levoj polovini neurona. Takav zbir je ulazni parametar za prenosnu funkciju neurona, obeleženu slovom f (na drugom crtežu su prikazane uobičajene prenosne funkcije). Izlaz iz neurona je samo jedan, a može da se vodi na više neurona u sledećem „sloju“ mreže.

## *Slojevi mreže*

Neuronska mreža se sastoji od većeg broja ovakvih neurona, koji su organizovani u dva ili više slojeva. Prvi sloj je uvek ulazni, a poslednji izlazni. Ako ih ima više, onda se unutrašnji slojevi zovu skriveni slojevi. Moguće je realizovati matricu i sa samo dva osnovna sloja, ali se složenije funkcije uvek realizuju sa bar jednim skrivenim slojem.

Način organizacije ovih slojeva i njihovog povezivanja čini arhitekturu ili topologiju neuronske mreže. Ona je specifična za svaki zadatak (što važi i za prenosne funkcije pojedinih neurona), ali postoje i mreže koje su sposobne da se same organizuju.

Prikazan je šematski izgled neuronske mreže sa četiri ulaza i tri izlaza. Sastoji se od tri sloja, a u skrivenom sloju ima osam neurona. Ovo je takozvana nepovratna (feed forward) mreža, jer nema povratne sprege – tok podataka (obično numeričkih vrednosti) je samo u jednom smeru, ali treba znati da postoje i povratne (feedback) mreže. Isto tako, mreže mogu da imaju i veći broj slojeva, da neke slojeve čini samo jedan neuron, a neke veoma veliki broj. Treba imati u vidu da je ovo pojednostavljena hipotetična mreža, i da u nekim slučajevima ona može da bude daleko složenija – recimo, da ima onoliko ulaza koliko piksela ima neka slika i onoliko izlaza koliko različitih motiva treba da prepozna.

## *Postupak učenja*

|  |
| --- |
| [http://pc.pcpress.rs/arhiva/images/broj/175/100_slika02.jpg](http://pc.pcpress.rs/arhiva/images/broj/175/100_slika02-big.jpg) |
|  |

Svakoj od strelica (izuzev izlaznih) dodeljen je težinski koeficijent, a svaki neuron ima svoju prenosnu funkciju i takozvani nivo aktivacije, koju čini promenljiva koja je deo te funkcije. Proces učenja (treniranja) mreže svodi se na to da se ulazi pobuđuju (recimo, mreži se „pokazuju“ bit‑mape slika ili slova), pri čemu će ona na izlazima pokazivati neke numeričke vrednosti, npr. verovatnoću da je na toj slici to što treba da prepozna.

Nepovratne mreže u postupku učenja koriste takozvani algoritam sa povratnim prostiranjem (back propagation), gde mreža najpre u normalnom smeru (sleva nadesno) generiše neki izlaz, pa onda u suprotnom smeru svakom neuronu dodeljuje količinu greške (razliku između željenog i realnog izlaza), pa u trećem prolazu (ponovo sleva nadesno) koriguje težinske koeficijente vodova koji pobuđuju taj neuron.

Počinje se od nasumično postavljenih vrednosti za težinske koeficijente. Tako „neiskusna“ mreža ne može ništa da prepozna. Trening teče tako što korisnik zadaje ulazne i željene izlazne vrednosti, dok se ne podese težinske koeficijente tako da se više ne menjaju. Onda joj se promene ulazi (recimo, pokaže druga slika ili slovo), pa se ponovi postupak učenja. Ovo se ponavlja za svaki ulazni podatak, posle čega se težinski koeficijenti više ne menjaju – svaki od neurona ih ima u svojoj memoriji, i mreža je spremna za rad. Mreža, dakle, u početku pravi određene greške, a odgovarajućom promenom težinskih koeficijenata smanjuje se razlika dobijenih i zadatih izlaznih vrednosti svakog neurona.

Važan parametar je nivo korekcije težinskih koeficijenata, jer ako je posle svakog koraka treninga taj nivo suviše mali, učenje će trajati predugo, a ako je veliki, težinski koeficijenti „podivljaju“ pa se učenje obično prekine zbog prekoračenja opsega aritmetike računara. Praksa pokazuje da je najbolje početi sa malim koeficijentima, postepeno ih povećavati jer će tokom učenja korekcije biti sve manje, a onda ih opet smanjivati kako bi se vrednosti težinskih koeficijenata fino podesile.

## *Praktična realizacija mreže*

Za razliku od mikroprocesora, neuronska mreža nije sekvencijalna, nema sistemski takt i u toku rada nije usredsređena na određene resurse, nego se podaci uniformno prostiru po celoj mreži, koja je protočna u realnom vremenu. Ako jedan deo mreže ispadne iz funkcije, to neće ozbiljno ugroziti njen rad (kao što bi ugrozilo rad mikroprocesora), a kod samoorganizujućih mreža bi drugi delovi čak mogli da preuzmu funkciju dela koji je van pogona i da odmah počnu sa učenjem, bez prekida u radu.

|  |
| --- |
| [http://pc.pcpress.rs/arhiva/images/broj/175/100_slika03.jpg](http://pc.pcpress.rs/arhiva/images/broj/175/100_slika03-big.jpg) |
|  |

Pored diskretnih postoje i analogne neuronske mreže, ali se one praktično i ne primenjuju. Neki proizvođači eksperimentišu sa mrežama na čipu, međutim to je još u eksperimentalnoj fazi, tako da se mreže još uvek simuliraju digitalnim kompjuterima.

Mada se neuronske mreže danas već uveliko koriste u procesnim upravljanjima, prepoznavanju teksta i likova, prognozi vremena i u sličnim domenima, treba imati u vidu da je ova tehnologija još uvek u fazi razvoja i da su sve to tek početni rezultati. Ako naše znanje o njima bude napredovalo onim tempom kojim su se razvijali digitalni kompjuteri, onda nas čekaju dramatične promene u tehničkom okruženju. Do sada je ovaj razvoj bio relativno spor, ali postoje optimistične prognoze da će veštačke neuronske mreže jednom moći da zamene ljudski mozak, jer su one sposobne da obavljaju zadatke, a da mi čak i ne znamo kako.

Neuronske mreže koje je priroda tokom evolucije ugradila u naše nervne sisteme tako su složene da je malo verovatno da ćemo uskoro moći da im konkurišemo. Tim pre što je za njihov razvoj priroda imala milijarde godina evolucije na milijardama jedinki, a mi ne možemo da dopustimo sebi taj luksuz. Osim toga, treniranje jedne prirodne neuronske mreže kod čoveka traje oko 20 godina, a to je predugo za eksperiment sa neizvesnim rezultatom.

Sledećeg meseca ćemo grubo opisati konkretan slučaj prepoznavanja teksta uz pomoć neuronske mreže (što svi postojeći OCR programi već uveliko koriste) i pozabaviti se problemima koje treba očekivati sa novim registarskim tablicama koje su, kako njihovi tvorci tvrde, dobro prilagođene mašinskom čitanju, automatskom registrovanju prekršaja i automatizovanom slanju kazni na kućnu adresu.

Radi se sa TensorFlow,Keras,Python,Anaconda,Jupyter,Spyder

**DEO 1 : dense i convent+dense**

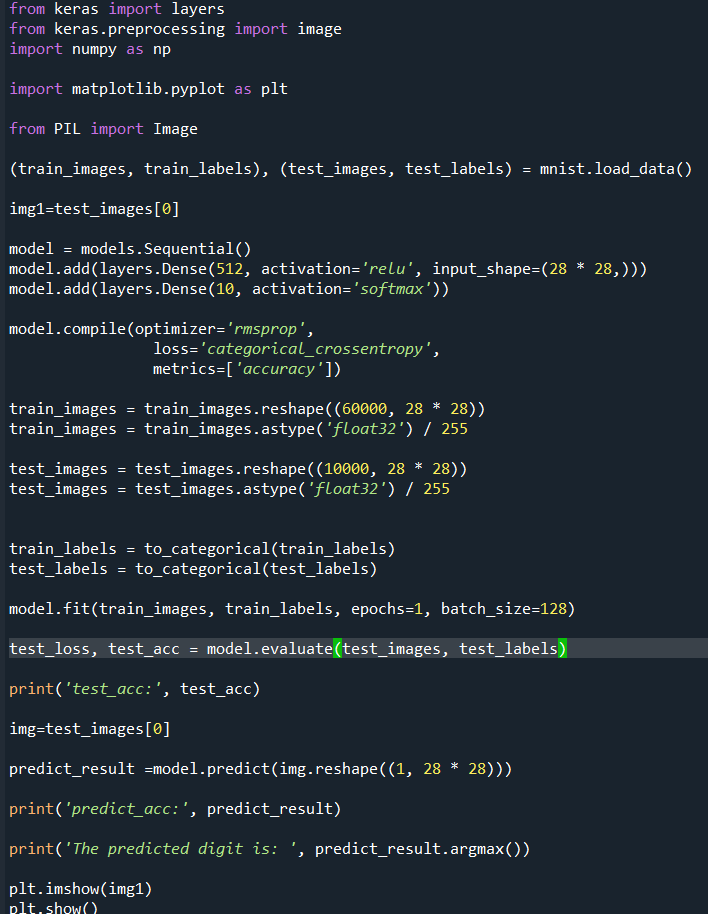
Koristi se Keras mnist dataset za prepoznavanje brojeva. Počinje se od dense mreže a zatim se dodaje konvolutivna mreža. Pokazujemo da je veci accuracy kada se uvede konvolutivna

mreza. Objašnjenji su osnovni pojmovi kao sto su trening , evaluacija i predikcija.

**Fajlovi :**

**1.Dense.py**

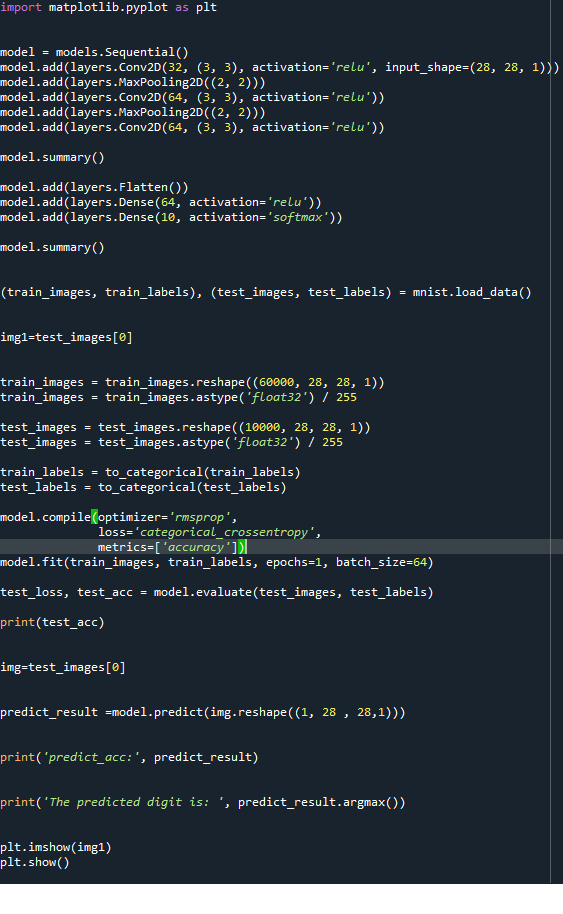
Primer gde je objasnjena Dense mreža sa fit,evaluate,predict. Nakon ucitavanja mnist dataset-a radimo predprocesuiranje slike, nakon cega sledi fit, evaluaciju I predikciju mreze. Kod je prikazan na Slika 1.



**Slika 1.**

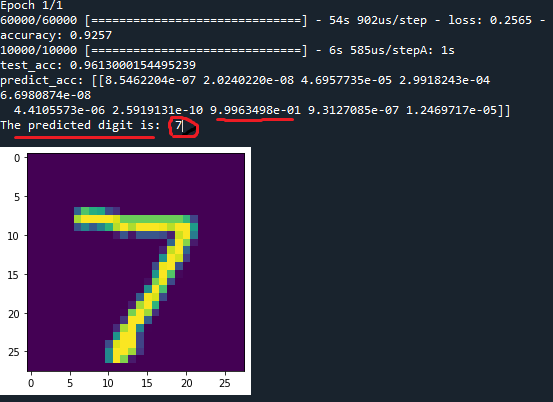
**1.Dense\_Convnet.py**

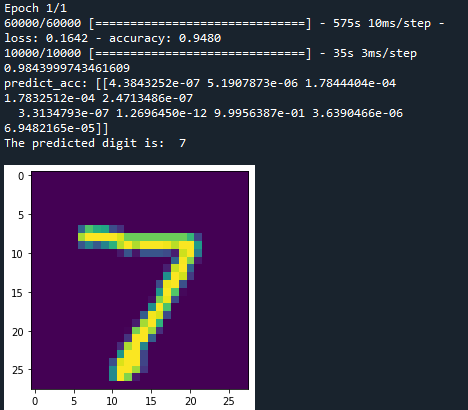
Primer gde je na prethodnu mrežu dodata Konvolutivna mreža. U primeru je povecavan accuracy u odnosu na prethodnu mrezu. Takodje je pokazano upotreba epoha i batch\_size-a. Na Slika 2 je prikazan kod.



**Slika 2.**

Slika 3. predstavlja uporedne rezultate rada dve prethodne mreze, gde se jasno vidi uvecanje accuracy.





**Slika 3.**

**DEO 2 : konvolutivne mreže za dataset dog\_cat**

Dataset link cats\_dog :

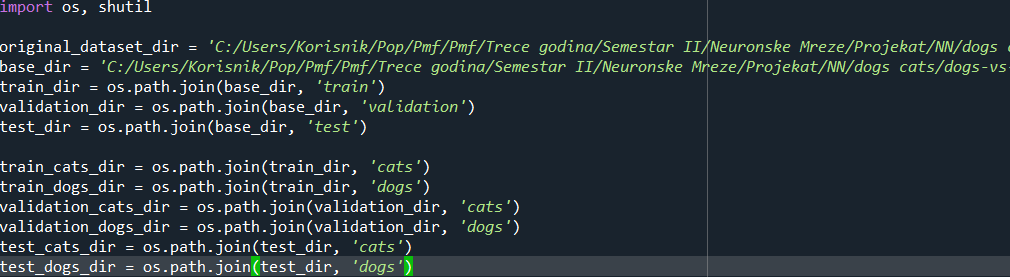
<https://drive.google.com/file/d/1XioMs_2ee7wB_yx1YIBmDiMNr4mMhdI5/view?usp=sharing>

Zbog lakseg testiranja i ucenja koristimo manji dataset od 4000 slika(2000 dog, 2000 cat). Na manjem setu cemo pokazati problem overfittinga i resavanje problema koji nastaju na manjim setovima. Data set delimo na 2.000 slika za trening,1.000 za validaciju i 1.000 za test.

**Fajlovi :**

**1.Constants.py**

Konstante koje se koriste u drugim fajlovima.Putanje do fajlova i direktorijuma. Prikazano na Slika 4.



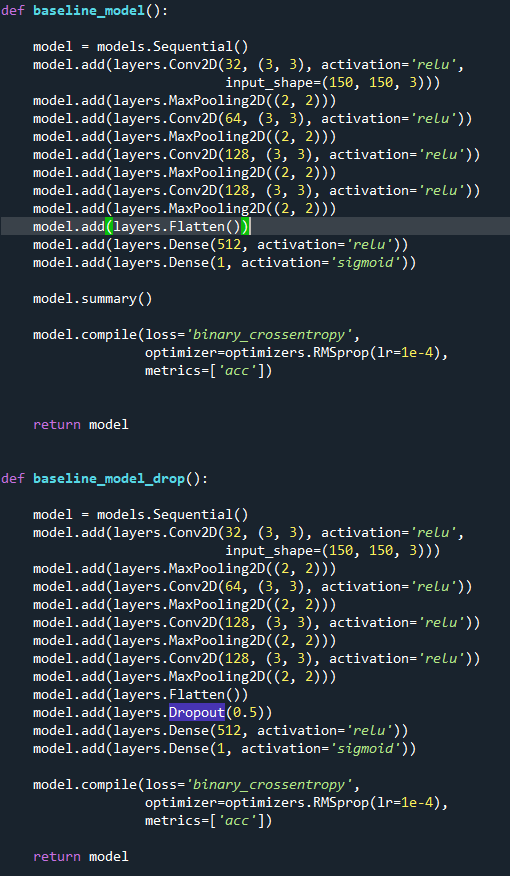
**Slika 4.**

**2.Buiding\_model.py**

Kreiranje modela.Modeli sa i bez Dropout. Prikazano na Slika 5.

Kreirana je konvolutivna mreza u kojoj je koriscen MaxPool. Ideja maxpoola je ocuvanje prostorne hijerarhije, cime se ocuvavaju medjusobni odnosi na slici. Tehnika se zasniva na tome da se pomocu kernel-a uzimaju maksimalne vrednosti piksela.

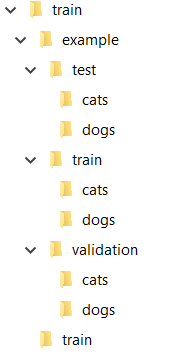
Dropout je random izbacivanje cvorova u mrezi tj. neurona.



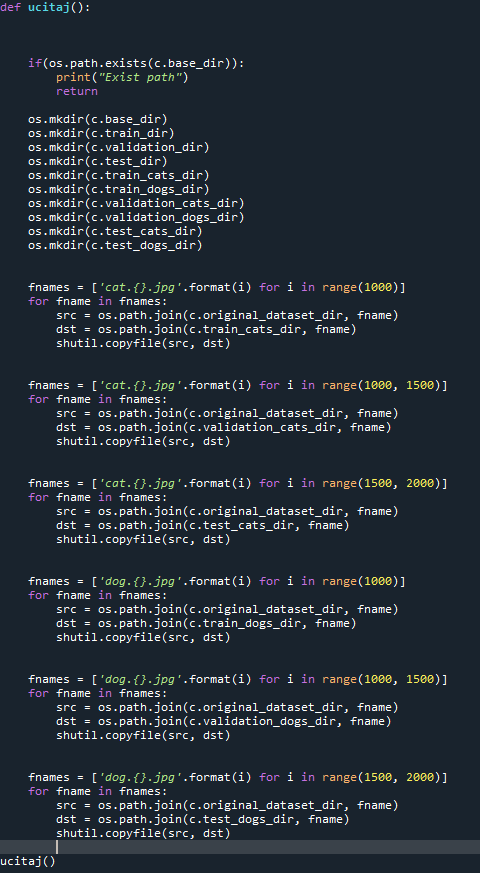
**Slika 5.**

**3.Create\_dataset\_folder.py**

Iz dataset-a cat\_dog formiraju se folderi za fit,train,validation uzimajuci odgovarajući broj slika. Struktura foldera na Slika 6,a kod je prikazan na Slika 7.



**Slika 6.**

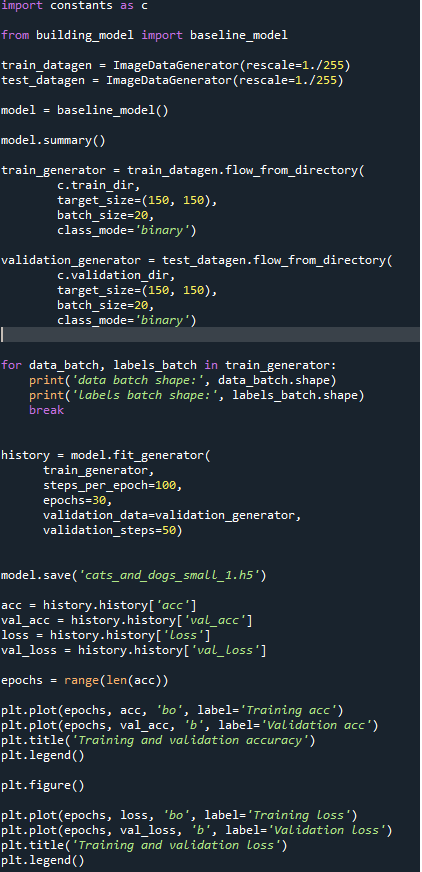


**Slika 7.**

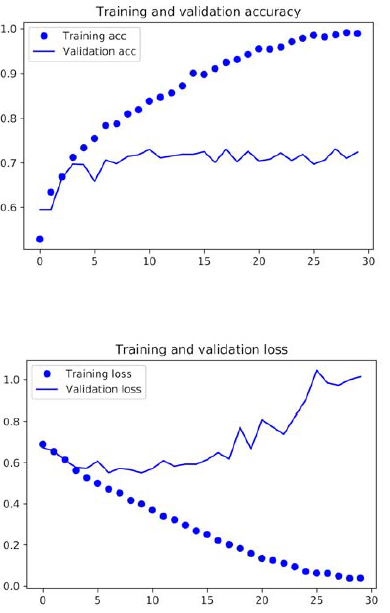
**4.Network\_1.py**

Mreža sa grafičkim prikazom Training acc i loss.problemi overfitting.

Formirana je konvolutivna mreza koriscenjem ImageGenerator-a sa upotrebom skaliranja. Pokazano je kako se model snima. Kod je prikazan na Slika 8, a graficki rezultati validation i accuracy na Slika 9.



**Slika 8.**

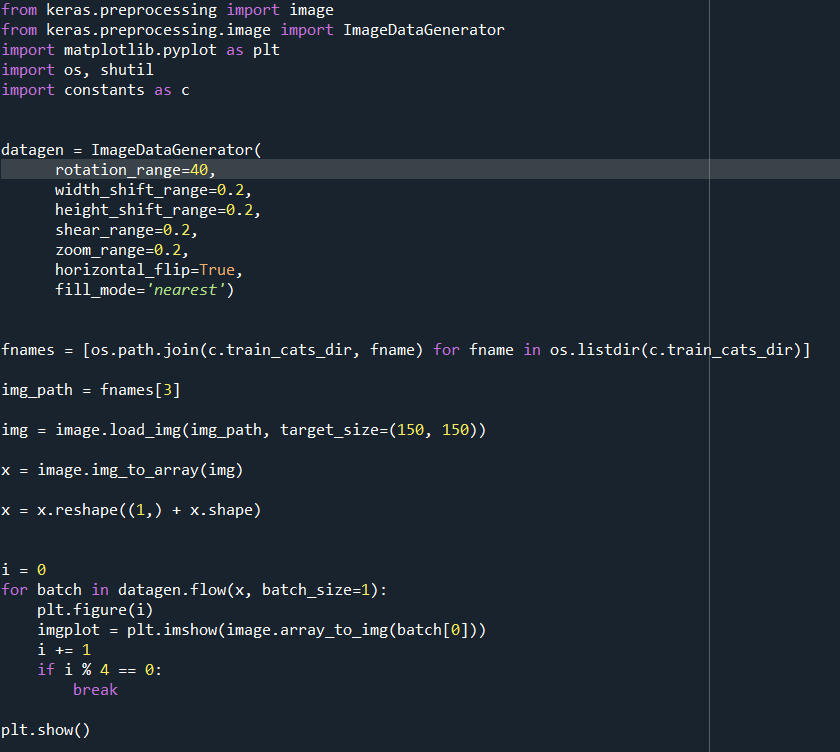


**Slika 9.**

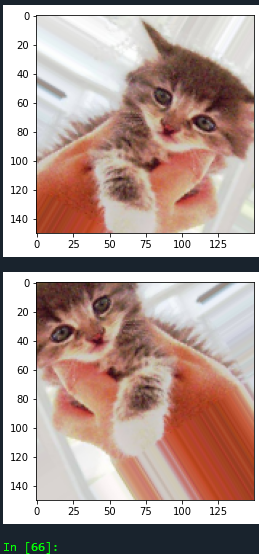
**5.Augmentation\_example.py**

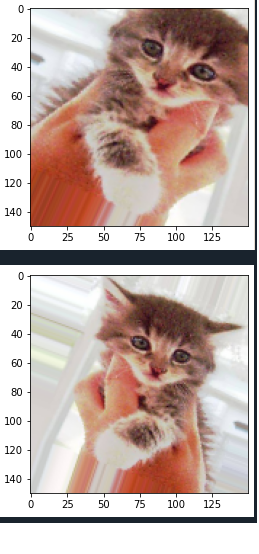
Primer augmentation na jednoj slici i nekoliko efekata (rotation,flip,zoom…). Kod je prikazan na Slika 10.

Pokazana je tehnike augmentacije na primeru jedne slike koriscenjem nekoliko efekata. Rezultati programa su prikazani na Slika 11.



**Slika 10.**

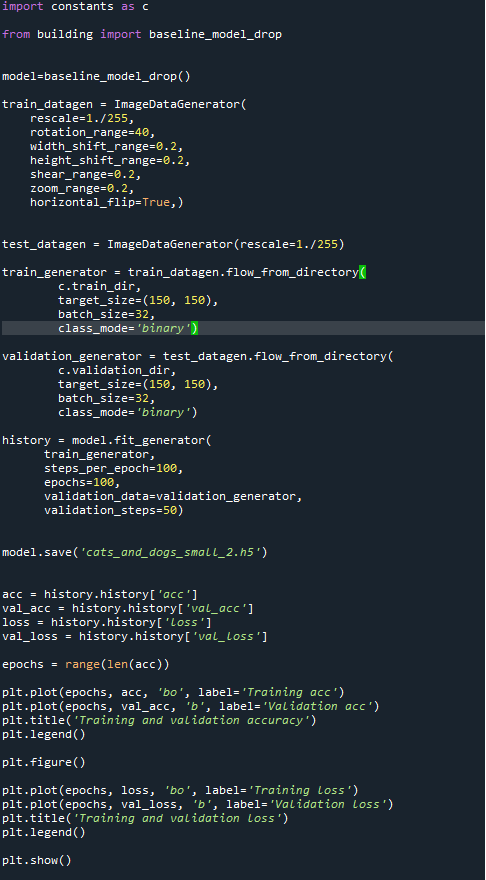




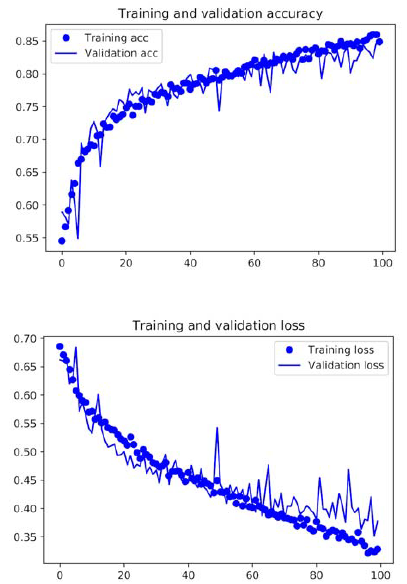
**Slika 11.**

**6.Network\_2.py**

Mreža koja je proširena u odnosu na Network\_1 sa augmentation i Dropout.Bolji rezultati su postignuti i kod overfitting. Prethodni model je prosiren augmentacijom i pokazano je da se ovom tehnikom dobijaju mnogo bolji rezultati. Kod je prikaza na Slika 12, a graficki prikaz rezultata na Slika 13.



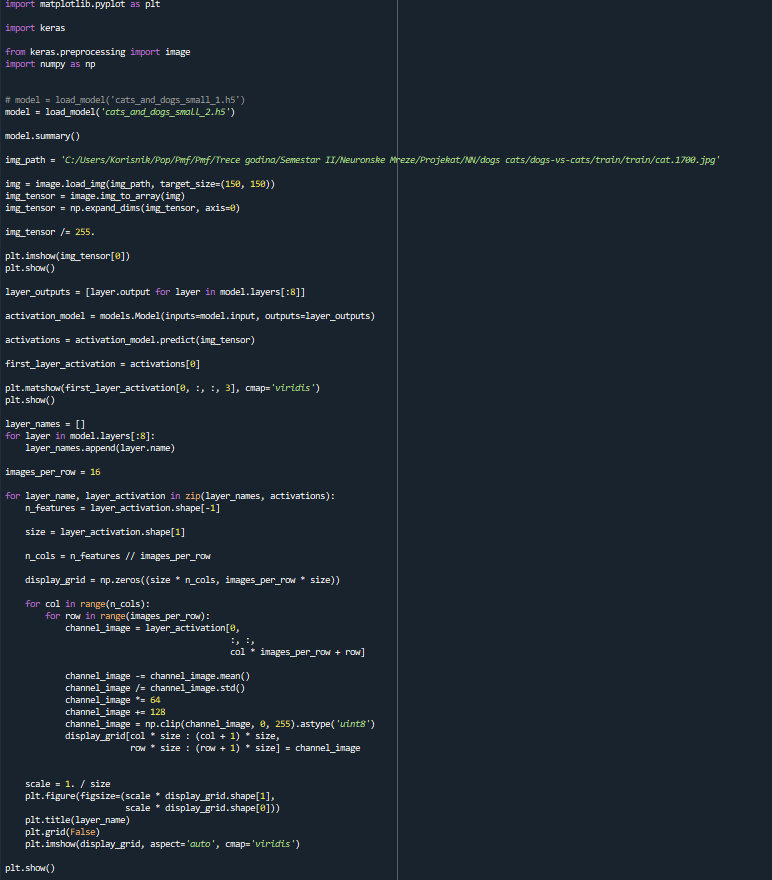
**Slika 12.**



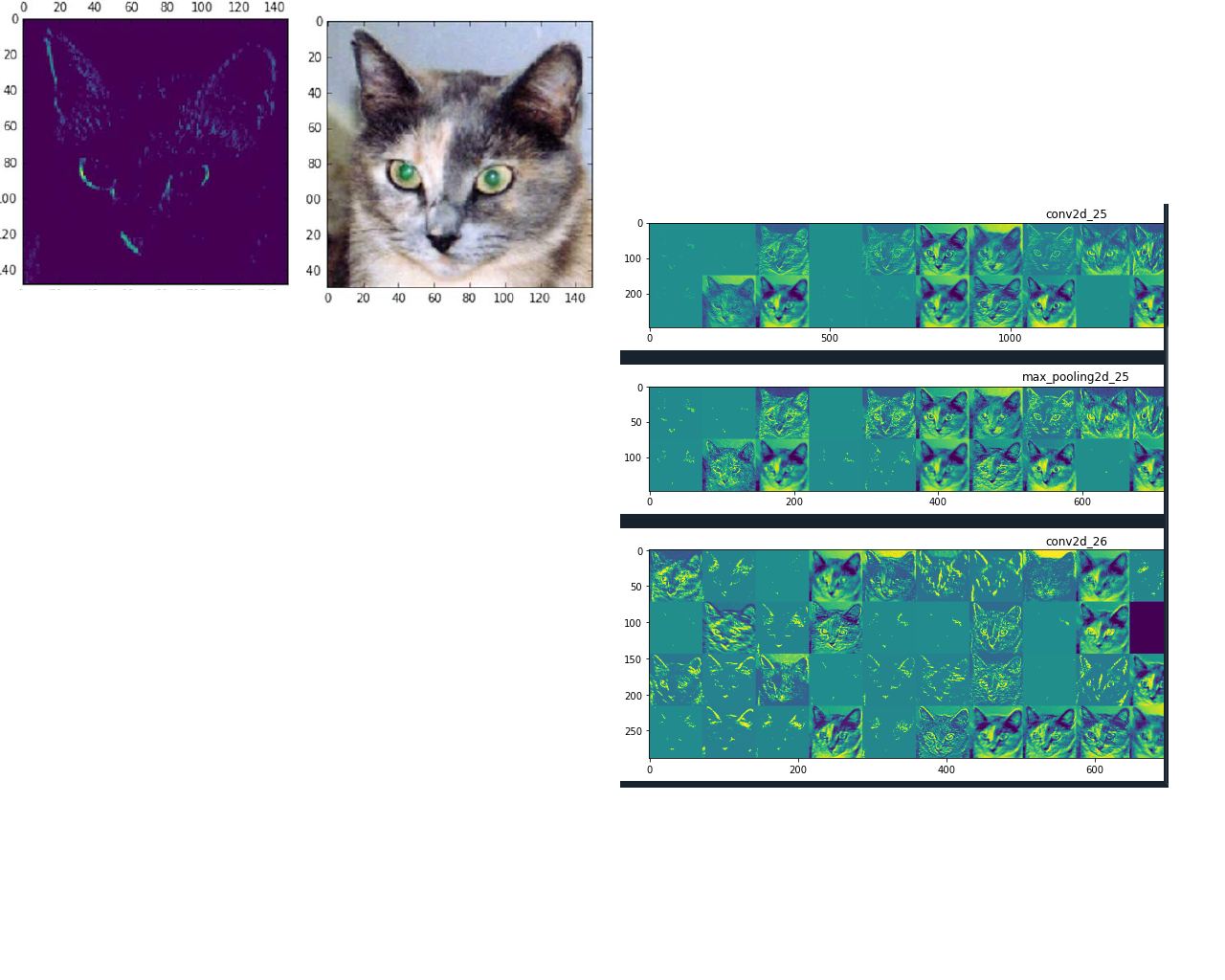
**Slika 13.**

**7.Vizualizing.py**

Vizualizacija intermedijarnih aktivacija sastoji se od prikazivanja features mapa koje se dobijaju na output-u različitih convolution i pooling slojeva, s obzirom na određeni ulaz (output layers se naziva i activation, to je output activation function). Ovo daje prikaz kako se ulaz razlaže u različite slike koje je naučila mreža. Svaki kanal ima nezavisne karakteristike, tako da je način vizuelizacije features mapa crtanje svakog kanala kao 2D slike. Krenuli smo od učitavanja modela koji smo sacuvali u prethodnim primerima.Kod je prikazan na Slika 14, a rezultat programa prikazan je na Slika 15.



**Slika 14**.



**Slika 15.**

**8.Vizualizing\_heatmap.py**

Ova vizuelizacije je korisna za razumevanje koji su delovi date slike doveli do konačne odluke o klasifikaciji , odnosno , koliko intenzivno ulazna slika aktivira određenu klasu.

Postupak se zastoji u tome da se odredi srednja vrednost(mean) poslednjeg sloja feature mapa(aktivacija) konvolutivnog dela mreze, u odnosu na ulaznu sliku. Tako određujemo koliko je svaki piksel feature mape važan za prepoznavanje sliku sa inputa tj. klasu kojoj pripada(dog,cat) .

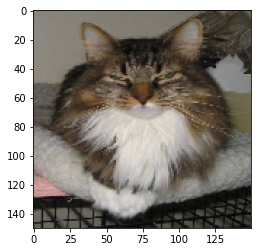
Pomenuta srednja vrednost svih feature mapa je heatmap aktivacija klase.

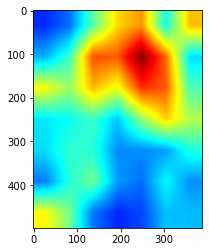
Drugim rečima , za convent dog-cat , ovo nam omogućava da generišemo heatmap za klasu cat, što ukazuje na to koliko različiti delovi slike izgledaju poput mačke, kao i heatmap za klasu dog koja označava delove slike nalik na pse.

Prikaz je na console a snimljen je i u fajlu.Takodje se koristi load modela iz cats\_and\_dogs\_small\_1.h5.

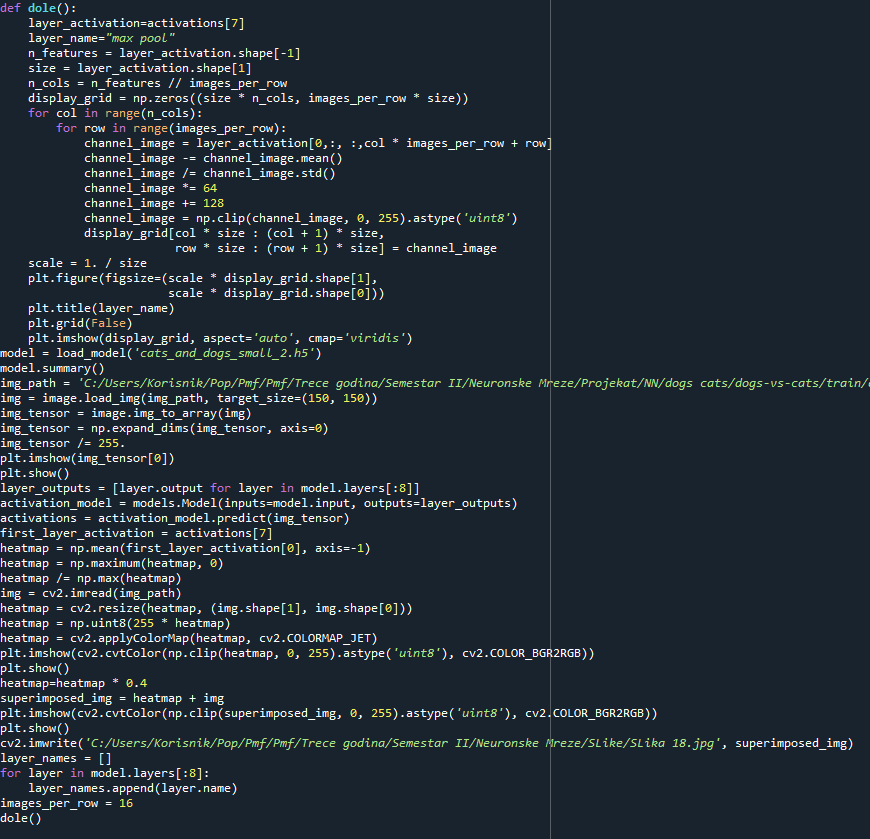
Slike za obe klase cat , dog su u nastavku.

Na Slika 16. prikazani su originalna slika, heatmapa, i rezultat spajanja. Kod je prikazan na Slika 17.



**Slika 16.**



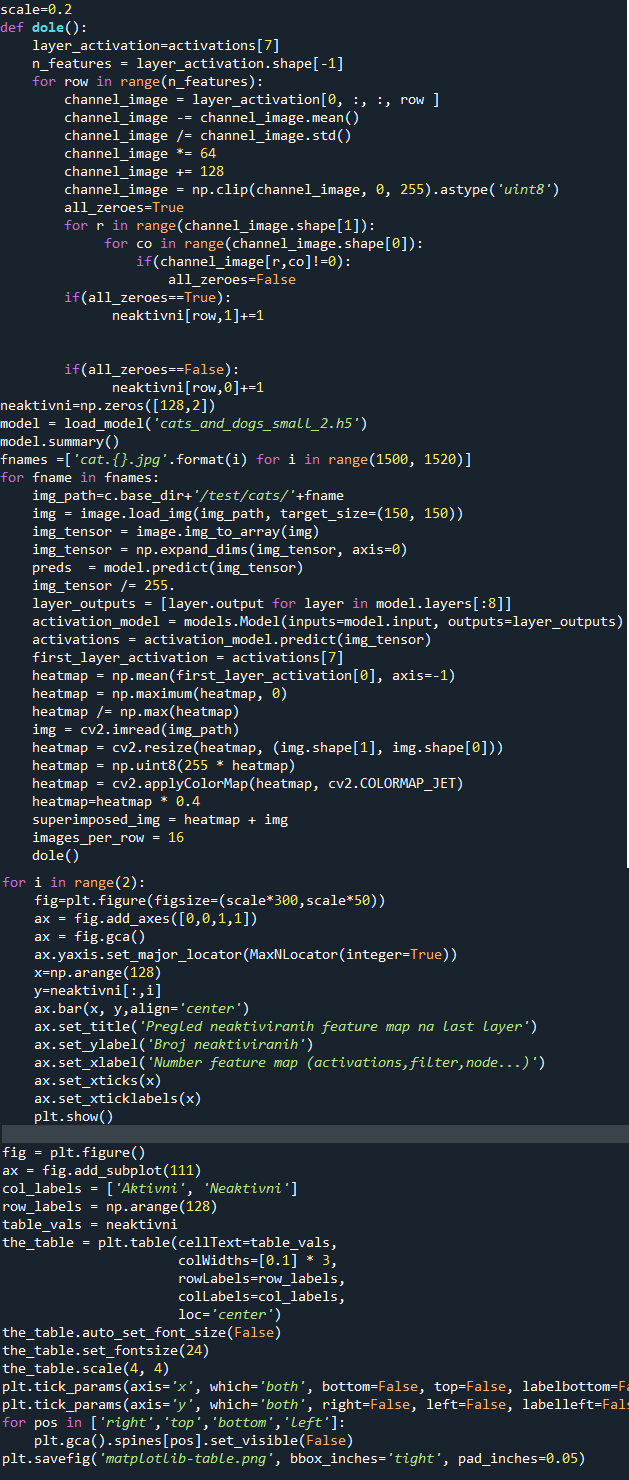
**Slika 17.**

**9.Vizualizing\_tabelarno.py**

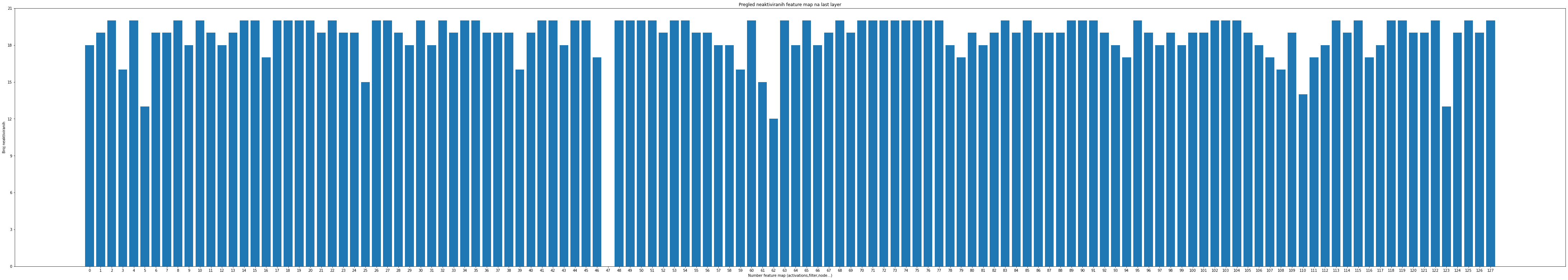
Ideje koje su razvijene u prethodnom kodu su iskoriscene za dobijanje podataka o tome koje feature mape(aktivacije) su se aktivirale u prepoznavanju odredjenih slika. Drugim recima za razlicite ulaze brojimo koliko puta su se odgovarajuce feature mape aktivirale.

Ideja je da feature mape posmatramo kao matrice brojeva i da nakon njihove normalizacije programski izbrojimo koliko je onih cije su sve vrednosti nula(nisu bitne za prepoznavanje), i onih cije vrednosti nisu nula(bitne su za prepoznavanje).

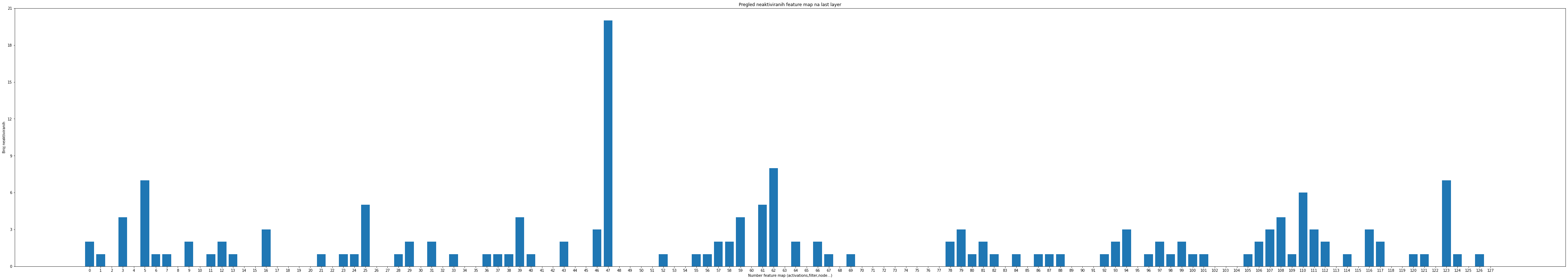
Kod je prikazan na Slika 18.Graficki i tabelarni prikaz prikazan je na Slika 19. Slika 20. Slika 21. Slika 22.



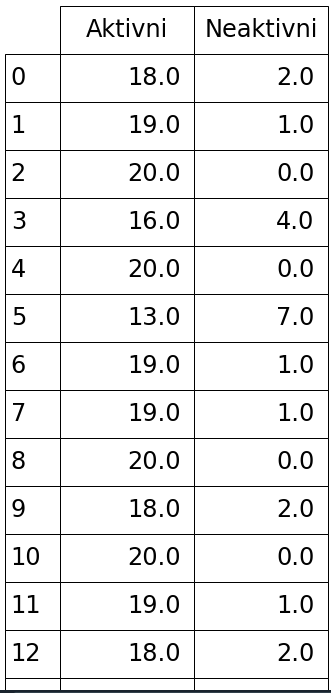
**Slika 18.**



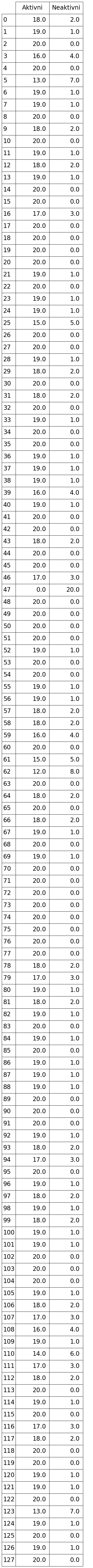
**Slika 19.**



**Slika 20.**



**Slika 21.**



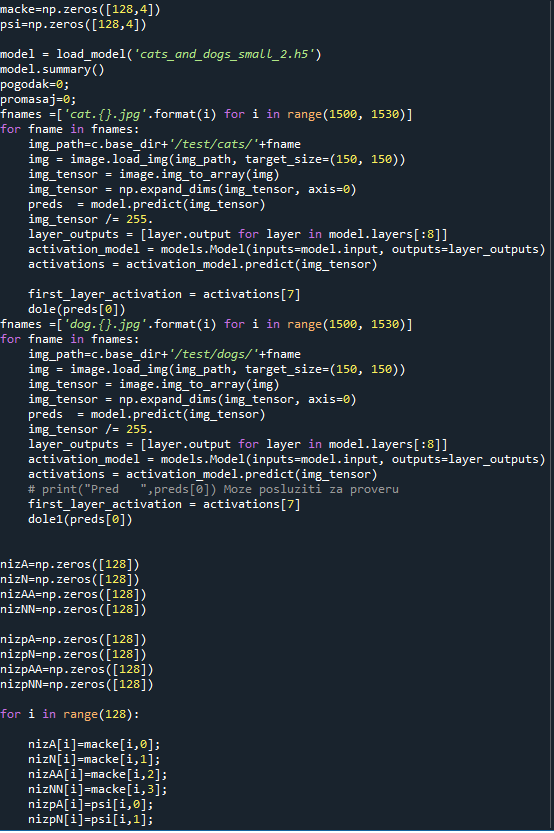
**Slika 22.**

**Dodatak:** Prikaz broja aktivacija u excel tabeli. Link je ispod, kao i slike.

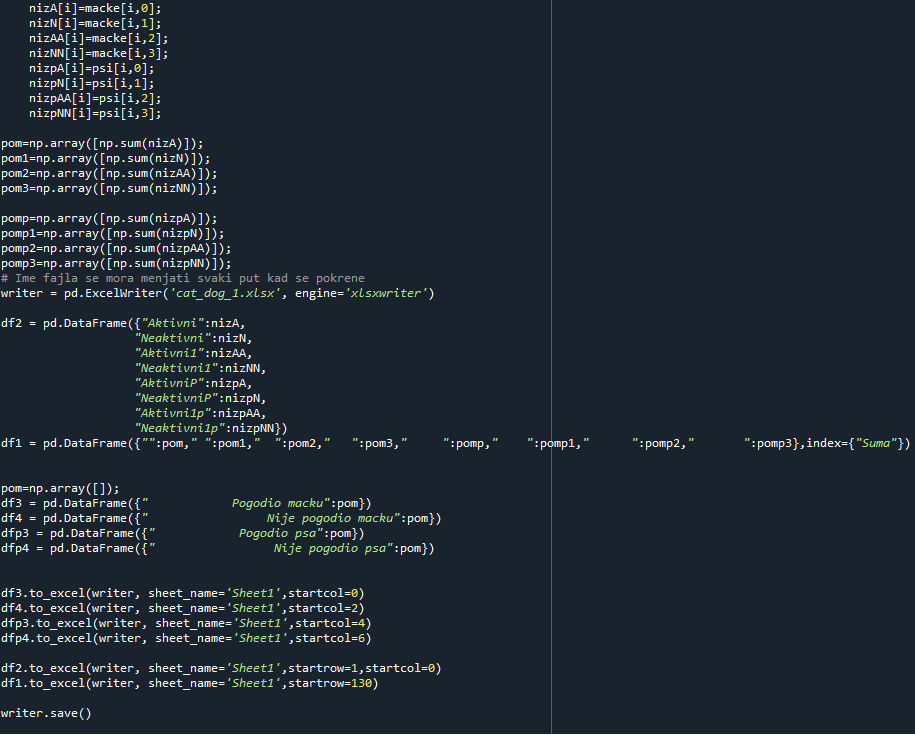
<https://drive.google.com/file/d/1h3T8pO6rs22YzlVymdSb7AUz5vFydvGU/view?usp=sharing>



**Slika 23.**



**Slika 24.**



**Slika 25.**