Generisanje i klasifikacija Pokémona po generacijama na osnovu postojećih Pokémona

Ana Anđelić
Fakultet Tehničkih Nauka
Novi Sad, Srbija
andjelic.r214.2023@uns.ac.rs

Teodor Sakal Francišković
Fakultet Tehničkih Nauka
Novi Sad, Srbija
sakal.franciskovic.r29.2023@uns.ac.rs

***Apstrakt*—Generative Adversial Networks (*GAN*-ovi) predstavljaju klasu neuronskih mreža korišćenih za nenadgledano učenje. Glavna uloga GAN-ova jeste generisanje veštačkih podataka koji teže da budu slični stvarnim podacima nad kojima su *GAN*-ovi trenirani. Prva generacija Pokémona je kreirana 1996. godine i od tada je ukupno kreirano devet generacija, koje uključuju 1025 različitih vrsta Pokémona. Ipak, u poslednjim generacijama se počela primećivati sličnost između starijih i novijih Pokémona. Kako bismo podstakli kreiranje raznovrsnijih Pokémona, istrenirali smo *Deep Convolutional GAN* (*DCGAN*) model nad augmentovanim podacima svih dosadašnjih Pokémona. Takođe smo istrenirali dodatnih devet *DCGAN* modela, po jedan za svaku generaciju, kao i model konvolucione neuronske mreže (eng. *Convolutional neural network, CNN)* koji nam služi za klasifikaciju Pokémona po generacijama. Ova studija slučaja je potvrdila da je moguće izgenerisati smislene Pokémone sa malim brojem epoha, kao i da postoje određeni šabloni (eng. *patterns*) u svakoj od generacija Pokémona, s obzirom da su nam rezultati klasifikacije visoki.**

***Keywords*— GAN, DCGAN, CNN, Pokémon, klasifikacija, augmentacija, veštački podaci.**

I. UVOD

Tokom prethodne decenije, mašinsko učenje je omogućilo generisanje različitih vidova lažnih podataka, kao što su tekstovi, slike, audio snimci i video klipovi. Stepen razvoja mašinskog učenja se ogleda u tome što je za prethodno spomenute lažne podatke u velikom broju slučaja teško primetiti da oni zaista jesu lažni. Generisanje lažnih slika se može videti na raznim sajtovima [1][2][3].

Pokémoni postoje još od 1996. godine, kada je izašla prva igrica u istoimenoj franšizi u dve verzije: *„Pokémon Red Version”* i *„Pokémon Blue Version”*. Ove igrice su obuhvatale originalnih 151 Pokémona, odnosno Pokémone koji pripadaju prvoj generaciji. Prethodno spomenuta franšiza je danas najprofatibilnija franšiza u industriji igara [4] i redovno nastavlja da proizvodi nove igrice. Poslednja generacija Pokémona koja je kreirana je objavljena krajem 2022. godine i deveta je po redu [5]. Iako je franšiza izuzetno uspešna, Pokémoni iz novijih generacija su počeli da liče na starije Pokémone. Takođe, u novijim generacijama je česta pojava da se stariji Pokémoni transformišu u alternativne verzije, sa obrazloženjem da oni proističu iz drugih regiona koji postoje u igricama. Potencijalno rešenje monotonosti novonastalih Pokémona može predstavljati generisanje novih slika Pokémona na osnovu prethodno postojećih Pokémona, pomoću metoda mašinskog, odnosno dubokog učenja.

Cilj ovog rada jeste da prikaže da je moguće generisati smislene i drugačije Pokémone metodama dubokog učenja, koristeći već postojeće Pokémone. Takođe, želimo da pokažemo da model dubokog učenja (konkretno, *CNN*) može detektovati šablone u novokreiranim generacijskim Pokémonima. Ove ciljeve postižemo tako što:

- kreiramo i treniramo *DCGAN* model, koji kao ulaz prima skup originalnih 1025 Pokémona (i njihovih alternativnih oblika), a za izlaz daje raznolik skup veštački-generisanih Pokémona;
- vršimo augmentaciju svih slika Pokémona, kako bi *DCGAN* modeli trenirani nad slikama Pokémona iz pojedinačnih generacija imali dovoljno ulaznih podataka;
- kreiramo i treniramo dodatnih devet *DCGAN* modela. Ulaz za svaki od njih je skup augmentovanih slika Pokémona iz svake generacije pojedinačno, a izlaz je skup veštački-generisanih Pokémona, koji bi trebalo da sadrže specifične šablone Pokémona iz te generacije;
- kreiramo i treniramo *CNN* model, čija je uloga da izvrši klasifikaciju prethodno generisanih Pokémona po generacijama. Evaluaciju ovog modela vršimo pomoću *F1 micro* mere.

Ostatak rada je organizovan na sledeći način. Poglavlje II. opisuje trenutno stanje u oblasti. U poglavlju III. se opisuju teorijske osnove potrebne za razumevanje rada. U poglavlju IV. se diskutuje o korišćenoj metodologiji u radu. Poglavlje V. diskutuje dobijene rezultate. Poglavlje VI. zaključuje rad.

II. STANJE U OBLASTI

Prvi koncepti koji se vezuju za *GAN*-ove se pojavljuju 2014. godine. Autori rada [6] upoređuju osnovnu konstrukciju estimacije generativnih modela sa *minimax* igrom za dve osobe. Proces treniranja opisuju kao paralelno treniranje dva modela: generativnog modela koji upija distribuciju podataka i diskriminativnog modela koji procenjuje mogućnost da je podatak iz originalnog trening skupa, a ne da je generisan od strane generatora. Zadatak generatora je da maksimizira šansu da diskriminator napravi grešku.

Nadogradnja *GAN*-ova se nastavljala u dolazećim godinama. Glavna poboljšanja novih modela se ogledala u stabilnosti i brzini modela. *DCGAN*-ovi koriste konvolucionalne slojeve radi kreiranja slika i autori koji su prikazali ovu arhitekturu konstatuju konvolucioni slojevi uče reprezentacioni hijerarhiju delova objekata kako iz generatora, tako i iz diskriminatora [7]. Autori koji su predstavili *Least Squares GAN* (*LSGAN*) konstatuju da *sigmoid cross entropy* funkcija gubitka može dovesti do

problema nestajućih gradijenata (eng. *vanishing gradient problem*) i zbog toga uvode da diskriminator koristi funkciju gubitka najmanjih kvadrata (eng. *least squares loss function*) [8].

Naredni korak u usavršavanju ovih struktura je predstavljao kreiranje slika visoke rezolucije. Ovo je postignuto pomoću arhitekture *Progressive GAN* (*PROGAN*), gde je osnovna ideja da se i generator i diskriminator postepeno povećavaju, odnosno da se dodavaju novi slojevi [9]. Godinu dana kasnije, dolazi do kreiranja arhitekture *BigGAN*, gde su autori primenili ortogonalnu regularizaciju (eng. *orthogonal regularization*) nad generatorom, trenirali model nad *ImageNet* 128x128 skupom podataka i uspeli da generišu slike veoma visokog kvaliteta [10].

Danas, unapređene verzije arhitektura *GAN*-ova se koriste u raznim industrijama, kao što su medicina, robotika, industrije gde je potrebno prepoznavanje prirodnog jezika (eng. *natural language processing*), itd. Trenutno popularna tema jesu *deepfake*-ovi, koji predstavljaju veštački kreirane slike, audio i video klipove neke osobe [11]. *Deepfake*-ovi su se razvili do te mere da nekada nije moguće raspoznati da li se radi o stvarnom ili veštački-generisanom sadržaju. Neke države su već donele zakone o tome da je upotreba *deepfake*-ova ilegalna, radi zaštite osoba koje se pojavljuju u tim sadržajima.

III. Teorijske osnove

U ovom poglavlju će biti prodiskutovane teorijske osnove pojmova potrebne za dalje razumevanje rada.

A. *Konvoluciona neuronska mreža*

Konvoluciona neuronska mreža je podtip neuronske mreže, koja se uglavnom koristi za sisteme gde je potrebno vršiti prepoznavanje slika i govora (eng. *image and speech recognition*). Konvolucioni slojevi smanjuju dimenzionalnost slika bez gubljenja bitnih informacija o istima. Zbog prisustva konvolucionih slojeva, konvolucione neuronske mreže daju bolje performanse u odnosu na standardne neuronske mreže, u slučaju kada je potreban rad sa slikama [12].

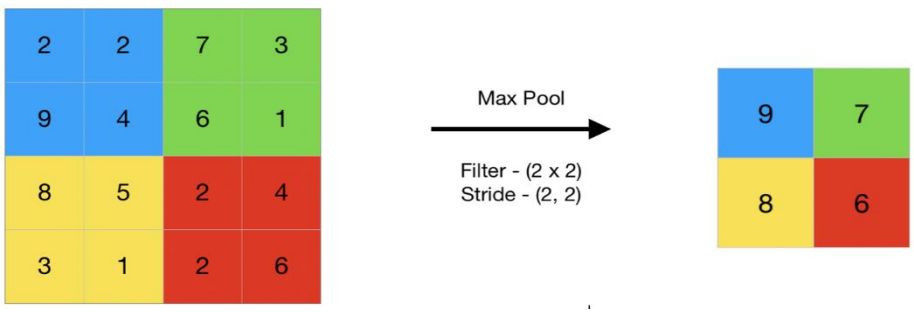
Novine, koje konvolucione neuronske mreže donose u odnosu na standardne neuronske mreže, su:

- konvolucioni slojevi (eng. *convolutional layers*): u konvolucionom sloju se preko ulaza (eng. *feature map*), odnosno slike koja je predstavljena kao matrica, prelazi filterom koji je manjih dimenzija nego ulazna matrica. Operacijom konvolucije između ulazne matrice i filtera se dobija matrica manjih dimenzija, koja sadrži najbitnije podatke vezane za ulaznu matricu. U slučaju da je slika u boji, potrebno je za svaku od komponenti boje (eng. *RGB – red, green, blue*) izvršiti konvoluciju sa zasebnim filterom. Takođe, moguće je definisati korak (eng. *stride*), koji određuje koliko piksela želimo preskočiti pre sledeće operacije konvolucije i zero padding, koji nam pomaže da ne izgubimo podatke vezane za sliku oko ivica slike, time što će dodati nule oko matrice nad kojom vršimo operaciju konvolucije.
- *pooling* slojevi (eng. *pooling layers*): uloga *pooling* slojeva je da redukuju dimenzije ulazne matrice i da uklone šum (eng. *noise*) sa slike. Potrebno je odrediti veličinu *pooling* operatora, koja je gotovo uvek veličine 2 x 2 [13]. Preko ulazne matrice se prelazi *pooling* operatorom i pronalaze se odgovarajuće vrednosti, koje se beleže u izlaznoj matrici. Postoje dva tipa *pooling* slojeva, a to su *average pooling*, gde je ideja da se uzme prosečna vrednost u svakom od delova matrice gde pooling operator prođe i *maximum* (*max*) *pooling*, gde je ideja da se uzme najveća vrednost u svakom od delova matrice gde pooling operator prođe. Moguće je definisati i korak *pooling operator*-a, koji ima identičnu ulogu kao i kod konvolucionih slojeva. Primer *max pooling* sloja, koji koristi *pooling* operator dimenzija 2 x 2 i korak (2, 2) je prikazan na slici 1.
- *flatten* sloj (eng. *flatten layer*) i potpuno povezani sloj (eng. *fully connected layer*): nakon što su izvučeni najbitniji detalji sa slike i smanjene dimenzije početne ulazne matrice, koristimo *flatten sloj*, čiji je cilj da transformiše ulaznu matricu u vektor. Novodobijeni vektor se dalje prosleđuje potpuno povezanom sloju, čiji je izlaz prediktovani rezultat za ulaznu sliku. Razlika u odnosu na standardne neuronske mreže je to što konvoluciona neuronska mreža sadrži potpuno povezani sloj samo na kraju svoje strukture.

B. *GAN*

GAN modeli su u poslednjoj deceniji postali popularni zbog mogućnosti generisanja i poboljšanja različitih tipova podataka. Arhitektura *GAN*-a je prikazana na slici 2, a neuronske mreže iz kojih se sastoji su:

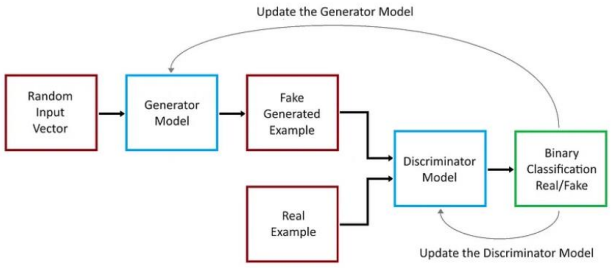
- generator. čija je uloga da generiše podatke nalik realnim podacima što je to više moguće. Ulaz u generator je nasumičan šum, dok je izlaz generisani podatak. Tokom treniranja, generator uči kako da kreira podatke tako da njih diskriminator ne može razlikovati. Ažuriranje težina (eng. *weights*) generatora se vrši na osnovu povratne informacije diskriminatora. Tipična arhitektura generatora se sastoji iz ulaznog, potpuno povezanih, izlaznog sloja i slojeva za izmenu oblika izlaznog podatka [14];
- diskriminator, čija je uloga da razlikuje stvarne i



Slika 1 - Primer max pooling sloja, koji koristi operator dimenzija 2 x 2 i korak (2, 2)

veštački-generisane podatke od strane generatora. Ulaz u diskriminator su podaci, koji mogu biti ili stvarni ili veštački-generisanim dok je izlaz verovatnoća da je ulazni podatak stvaran. Treniranje izgleda tako što su diskriminatoru pokazani stvarni i veštački-generisani podaci, gde za stvarne podatke na izlazu daje vrednosti blizu 1, a za veštački-generisane podatke daje vrednosti blizu 0. Tipična arhitektura diskriminatora je slična arhitekturi *CNN*-a. Funkcija gubitka kod diskriminatora je *binary cross-entropy*, s obzirom da je potrebno razlikovati stvarne od veštački-generisanih slika [14].

Funkcija gubitka za celokupan *GAN* model uzima u obzir funkcije gubitka generatora i diskriminatora. Ova funkcija gubitka se zove *minimax loss*, a kod nje diskriminator pokušava da poveća procenat tačno prediktovanih stvarnih i veštački-generisanih slika, dok generator radi suprotno [14].



Slika 2 - Arhitektura *GAN* modela [14]

C. *DCGAN*

DCGAN predstavlja podtip *GAN* modela, koji je usko specijalizovan za zadatke koji obuhvataju generisanje slika. Glavna razlika u odnosu na standardne *GAN* modele jeste arhitektura *DCGAN* modela, koja umesto potpuno povezanih slojeva sadrži transponirane konvolucione slojeve (eng. *transposed convolutional layers*), koji se još zovu i dekonvolucionalni slojevi (eng. *deconvolutional layers*). Uloga ovih slojeva jeste da odrade *upsampling* podataka (npr. konvertovanje slike niske rezolucije u sliku visoke rezolucije). Pored dekonvolucionih, postoje i *batch normalization* slojevi koji doprinose stabilnosti treniranja *DCGAN* modela [15].

IV. METODOLOGIJA

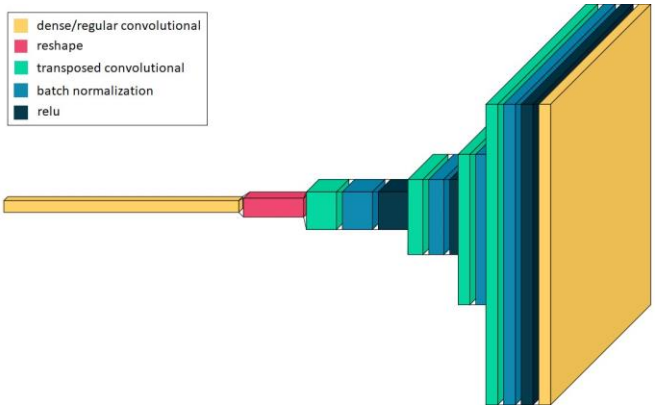
Izrada ovog projekta obuhvatila je konstrukciju i treniranje devet *DCGAN* modela nad određenom gneracijom Pokémona, jedan *DCGAN* nad svim Pokémonima, kao i *CNN* klasifikatora Pokémona nad podacima dobijenih iz devet specijalizovanih *DCGAN*-ova. Ovo poglavlje uključuje i pregled dve vrste augmentacije podataka, odnosno augmentacija čitavog skupa i augmentacija pri treniranju.

A. *Struktura DCGAN modela*

Struktura svih *DCGAN* modela je identična za svaku generaciju Pokémona, i kao što je prethodno rečeno, sastoji se iz generatora i diskriminatora. Arhitektura koja je opisana u sledećim potpoglavlju je inspirisana člankom autora Gonçalo Chambel-a [18]. Dijagrami arhitektura su generisani uz pomoć *visualkeras* [17] *Python* biblioteke. Prilikom njihovog treniranja su korištene *Adam* optimizaciona funkcija i *Binary Cross Entropy* kao funkcija gubitka.

1) Generator

Arhitektura generatora se može videti na slici 3. Cilj generatora je da iz nasumičnog ulaza generiše novu sliku. Prvi deo generatora predstavlja *fully connected* sloj koji povećava dužinu vektora sa 100 na 65536 (8*8*1024). Ovaj sloj prati *reshape* sloj koji ovaj jednodimenzioni vektor konvertuje u vektor dimenzija 8x8x1024.



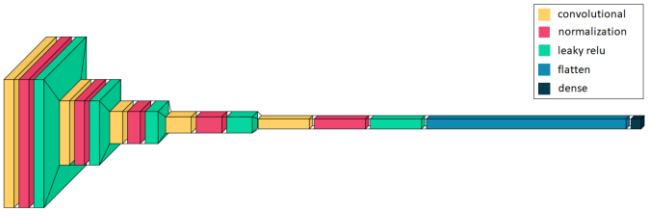
Slika 3 - Arhitektura generatora

Nakon toga slede četiri bloka koji uvećavaju širinu i visinu, a smanjuju dubinu ulaznog vektora, a jedan takav blok se sastoji iz jednog transponiranog konvolutivnog sloja, *batch* normalizacije i *ReLU* aktivacione funkcije. Za razliku od običnog konvolutivnog sloja koji izdvaja, transponirani konvolutivni sloj generiše karakteristike i uvećava širinu i visinu vektora, dok smanjuje njegovu dubinu. Dimenzije vektora pri ulazu u prvi ovakav blok je 8x8x1034, a pri izlazu iz poslednjeg 128x128x64. Veličina kernela kod svih transponiranih konvolutivnih slojeva je 5x5, a korak 2x2.

Poslednji sloj ove mreže je običan konvolutivni sloj koji ima veličinu kernela 5x5, korak 1x1, a broj filtera je tri kako bi se kao izlaz dobila slika sa tri kanala.

2) Diskriminator

Arhitektura diskriminatora se može videti na slici 4. Cilj diskriminatora je da za sliku predvidi da li je prava ili lažna. Sastoji se iz četiri konvolutivna bloka, *flatten* sloja i *fully connected* sloja.



Slika 4 - Arhitektura diskriminatora

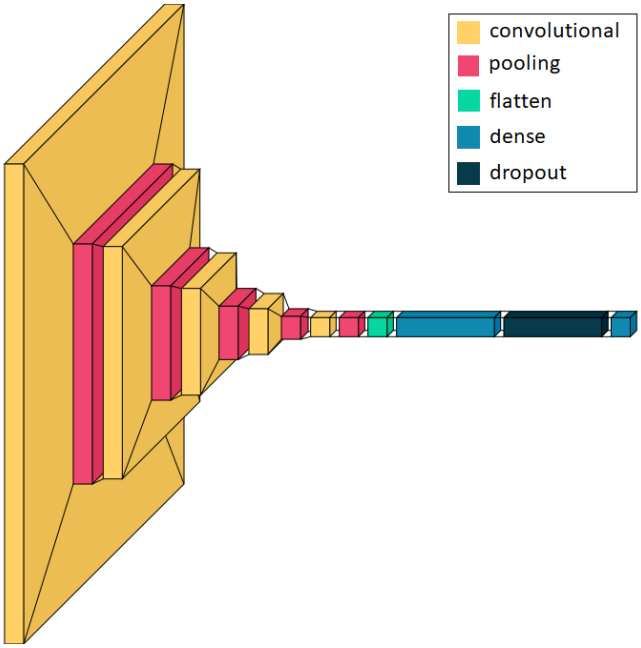
Konvolutivni blokovi služe za ekstrakciju karakteristika, i sastoje se iz konvolutivnog sloja, *batch* normalizacije i *Leaky ReLU* aktivacione funkcije. Vektor je pri ulazu u prvi konvolutivni blok dimenzija 128x128x3, dok je na izlazu iz poslednjeg konvolutivnog bloka dimenzija 8x8x1024.

Na kraju slede *flatten* i *fully connected* sloj. *Flatten* sloj pretvara višedimenzioni vektor u jednodimenzionalan, odnosno iz 8x8x1024 u 1x1x65536. Nakon toga *fully connected* sloj sa dužinom izlaza 1 daje konačnu predikciju

da li je slika koja je prosleđena diskriminatoru prava ili lažna.

B. *Struktura klasifikatora*

Ulazni sloj klasifikatora, prikazan na slici 5 prima *RGB* sliku dimenzija 128x128x3, a kao konačan izlaz vraća vektor dužine devet zbog devet generacija Pokémona koje su ovim projektom obuhvaćene.



Slika 5 - Arhitektura klasifikatora

Slojeve ove mreže na početku čine pet naizmeničnih konvolutivnih i *pooling* slojeva. Konvolutivni slojevi sadrže kernel veličine 3x3 i *ReLU* aktivacionom funkcijom, a broj filtera je redom 32, 64, 128, 64 i 32. Svi *pooling* slojevi sadrže kernel veličine 2x2.

Nakon konvolutivnih i *pooling* slojeva sledi *flatten* sloj. Ovaj sloj služi za konverziju ulaznog multidimenzionalnog vektora u jednodimenzionalni vektor.

Nakon *flatten* sloja slede *fully-connected* slojevi neuronske mreže. Prvi *fully-connected* sloj je dužine 1024 sa aktivacionom funkcijom *ReLU*, nakon kog sledi *dropout* sloj koji sledi nasumično postavlja vrednosti nekih elemenata na nula, što je korisno pri prevenciji preprilagođavanja. Na samom kraju imamo još jedan *fully-connected* sloj sa *softmax* funkcijom i dužinom outputa devet, koji će nam aktivacijom jednog od devet neurona dati konačan rezultat klasifikacije.

Prilikom kompajliranja ovog modela koristili smo *Adam* optimizacioni algoritam i *Categorical Cross Entropy* funkciju gubitka.

C. *Augmentacija čitavog skupa podataka*

Originalni skup podataka se sastojao iz 1120 slika, što je premalo za treniranje *DCGAN*-a, pa je stoga izvršena augmentacija podataka kako bi se skup veštački povećao. Augmentacija se vršila *ImageDataGenerator* klasom iz *TensorFlow*-a. Podešeni su sledeći parametri:

- *rotation_range=75* – slike se nasumično rotiraju za ugao u opsegu [-75, 75] stepeni;
- *brightness_range=(0.25, 1.75)* – nasumično se podesi svetlina slike iz datog opsega;
- *shear_range=45* – fiksira se jedna osa i slika se rasteže/rotira po drugoj osi za ugao u opsegu [-75, 75] stepeni;
- *zoom_range=[0.7, 1.3]* – slika se skalira u skladu sa slučajno odabranim brojem iz ovog opsega;
- *horizontal_flip=True* – slike će se nasumično okrenuti kao u ogledalu;
- *fill_mode='constant'* – pikseli će biti popunjeni konstantnom vrednošću koja se specificira pomoću argumenta *cval*;
- *cval=255* – vrednost koja će se koristiti za piksele koji se dodaju prilikom proširivanja slika.

D. *Augmentacija prilikom treniranja*

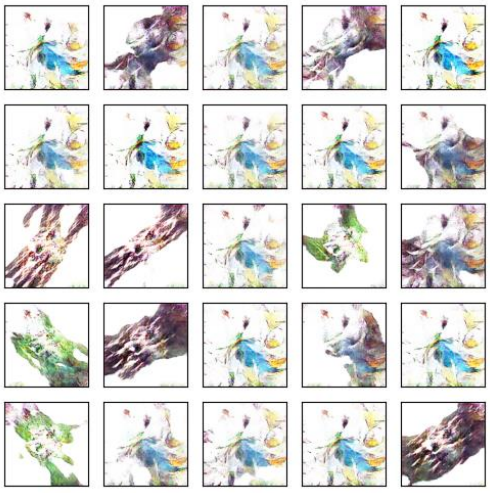
Nakon pokušaja treniranja modela nad podacima sa prethodno navedenom augmentacijom, nismo naišli na dovoljno dobre rezultate. Stoga smo tokom treniranja dodatno augmentovali svaki *batch* posebno tako što smo menjali osvetljenje, zasićenost, kontrast, translirali sliku i nasumično isecali deo slike [16]. Isecanje dela slike se vrši postavljanjem crnog pravougaonika nasumičnih dimenzija na nasumični deo slike. Više o doprinosu dodatne augmentacije će biti diskutovano u poglavlju V.

V. Rezultati

Svi *DCGAN*-ovi su trenirani na 400 epoha i pritom su sačuvani rezultati generatora nakon svakih 50 epoha, kao i grafik funkcije gubitka za sve modele. Klasifikator je treniran na 100 epoha nad slikama Pokémona generisanih specijalizovanim *DCGAN*-ovim.

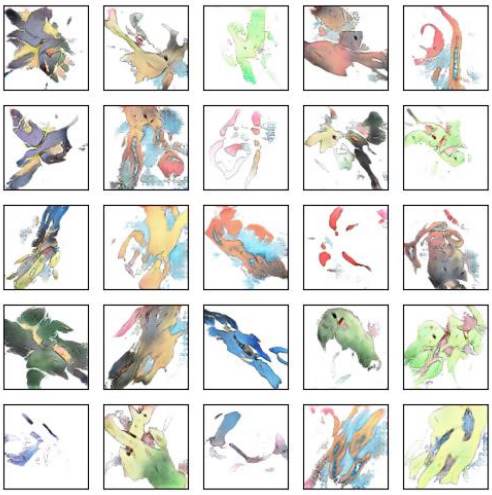
A. *Rezultat uvođenja dodatne augmentacije*

Prilikom treniranja *GAN*-ova se prvo koristio samo prvi oblik augmentacije, gde su slike augmentovane pre samog treniranja i neke od tehnika augmentacije su bili rotiranje, izmena osvetljenja, zumiranje, horizontalno okretanje. Rezultati generatora nakon 400. epohe se mogu videti na slici 6.



Slika 6 - Rezultat 400. epohe sa samo jednom augmentacijom

Da bismo dalje poboljšali rezultate, dodata je augmentacija tokom treniranja, gde je svaki *batch* povećan menjanjem osvetljenja, zasićenosti, kontrasta, transliranjem slike i isecanjem delova slike. Rezultate generatora nakon 400. epohe sa dodatnom augmentacijom možemo videti na slici 7.



Slika 7 - Rezultat 400. epohe sa dodatnom augmentacijom

Poređenjem ove dve slike primećujemo da je u prvom primeru generator pronašao „šablon“ koji diskriminator loše klasifikuje, dok dodatnom augmentacijom dobijamo više različitih Pokémona i bolji kvalitet slike. Zbog ovoga se za sve *DCGAN*-ove dalje koriste obe vrste augmentacije.

B. *Performanse DCGAN-ova*

Prilikom treniranja *DCGAN*-ova pratili smo generisanje slika na svakih 50 epoha i na kraju iscrtavali grafik funkcije gubitka za diskriminator i generator. Tokom treniranja se diskriminatoru smanjuje, dok se generatoru povećava funkcija gubitka. Na slici 8 možemo videti funkcije gubitka za sve modele koje smo trenirali.

Kao što se na graficima može primetiti, vrednosti funkcije gubitka zavise od generacije do generacije. Najniže vrednosti možemo uočiti kod druge generacije, gde diskriminator dostiže vrednost funkcije gubitka ispod 1, a generator do 10. Najveće vrednosti možemo primetiti kod pete generacije gde diskriminator pri kraju treniranja dostiže približno 25, a generator čak i do 300 za vrednost funkcije gubitka.

Funkcije gubitka na kraju treniranja još uvek nisu stabilno iskonvergirale, što bi se postiglo nastavkom treniranja. Zbog ograničenja resursa za treniranje ovo možemo ostaviti kao priliku za unapređenje u sledećem istraživanju.

C. *Performanse klasifikatora*

Klasifikator je treniran nad 18000 slika generisanih iz devet specijalizovanih *DCGAN*-ova. Metrike koje je klasifikator postigao na testnom skupu se mogu videti u Tabela 1. Zbog većeg broja klasa se za *precision*, *recall* i *F1* meru koristi macro parametar. Testni skup sadrži po 400 novih slika Pokémona generisanih od strane generatora dobijenih iz *DCGAN*-ova koji su trenirani nad posebnim generacijama.

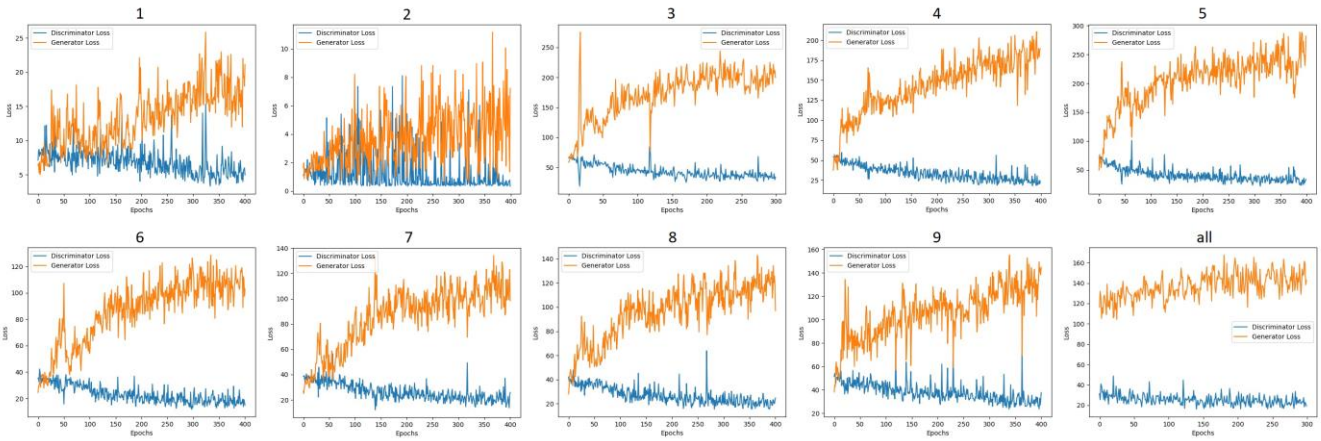
accuracy	*precision*	*recall*	*F1*
0.9511	0.9516	0.9511	0.9511

Tabela 1 - Metrike klasifikatora nad testnom skupu

Nakon evaluacije, klasifikator je testiran nad slikama generisanim generatorom koji je treniran nad svim Pokémonima. Ovde je primećeno da klasifikator, uprkos tome što je postigao dobre metrike prilikom evaluacije, ima tendenciju da većinu slika klasifikuje klasom dva ili šest. Ova pojava se može videti na slici 9.

Slika 9 - Rezultat klasifikacije slika generisanih od strane generatora koji je treniran nad svim generacijama

S obzirom da je klasifikator postigao dobre metrike prilikom treniranja, ovo može biti rezultat nepoklapanja



Slika 8 - Grafici funkcija gubitka svih *DCGAN*-ova

trening i krajnjeg testnog skupa podataka.

VI. Zaključak

Svrha ovog rada je da prikaže kako se treniranjem *DCGAN* modela nad manjim brojem epoha, mogu dobiti relativno kvalitetni rezultati. Mi smo se odlučili da ovaj cilj ostvarimo nad tematikom pokemona. Kako bismo podstakli kreiranje raznovrsnijih pokemona istrenirali smo deset različitih *DCGAN* modela. Takođe smo istrenirali i klasifikator koji generisane pokemone klasifikuje u jednu od devet klasa, radi evaluacije istreniranih *DCGAN*-ova. Dobijeni rezultati su sledeći:

- korišćenje dodatne augmentacije tokom treniranja je poboljšalo raznovrsnost i kvalitet generisanih slika;
- funkcije gubitka *GAN*-ova ne konvergiraju pri kraju treniranja, što znači da bi nastavkom treniranja postigli još bolji rezultat;
- klasifikator dostiže odlične mertike prilikom evaluacije nad podacima generisanih od strane specijalizovanih *GAN*-ova, dok ima tendenciju da podatke izgenerisane *GAN*-om svih pokemona klasifikuje u generacije dva i šest.

U budućem radu bismo unapredili arhitekture *DCGAN*-ova i trenirali ih nad više epoha kako bismo dobili još kvalitetnije slike pokemona.

VII. References

[1] https://generated.photos/, datum i vreme pristupa: 26.3.2024. 12:02h

[2] https://www.realfakephotos.com/, datum i vreme pristupa: 26.3.2024. 12:03h

[3] https://openai.com/gpt-4, datum i vreme pristupa: 26.3.2024. 12:04h

[4] https://www.titlemax.com/discovery-center/lifestyle__trashed/the-top-50-highest-grossing-video-game-franchises/, datum i vreme pristupa: 26.3.2024. 12:10h

[5] https://scarletviolet.Pokémon.com/en-us/, datum i vreme pristupa: 26.3.2024. 12:15h

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Networks", 2014.

[7] Alec Radford, Luke Metz, Soumith Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", 2016.

[8] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, Stephen Paul Smolley, "Least Squares Generative Adversarial Networks", 2017.

[9] Samuli Laine, Tero Karras, Jaakko Lehtinen, Timo Aila, "Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2018.

[10] Andrew Brock, Jeff Donahue, Karen Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis", 2019.

[11] https://www.techtarget.com/whatis/definition/deepfake, datum i vreme pristupa: 26.3.2024. 17:52h

[12] https://towardsdatascience.com/using-convolutional-neural-network-for-image-classification-5997bfd0ede4, datum i vreme pristupa: 29.3.2024. 16:13h

[13] https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/, datum i vreme pristupa: 29.3.2024. 17:03h

[14] https://medium.com/@marcodelpra/generative-adversarial-networks-dba10e1b4424, datum i vreme pristupa: 29.3.2024. 17:30h

[15] https://www.geeksforgeeks.org/difference-between-gan-vs-dcgan/, datum i vreme pristupa: 30.3.2024. 14:27h

[16] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, Song Han, " Differentiable Augmentation for Data-Efficient GAN Training", 2020.

[17] visualkeras · PyPI, datum i vreme pristupa: 4.4.2024. 10:17h

[18] Generating realistic Pokemons using a DCGAN | by Gonçalo Chambel | Medium, datum i vreme pristupa: 4.4.2024. 18:36h