Generiranje melodija korištenjem strojnog učenja

Margan, Tino

Sveučilište u Rijeci - Fakultet informatike i digitalnih tehnologija

Diplomski jednopredmetni studij informatike - modul PI

Tino Margan

Generiranje melodija korištenjem strojnog učenja

Diplomski rad

Mentor: doc. dr. sc. Miran Pobar

Rijeka, 2022.

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

Rijeka, 5.6.2022.

Zadatak za diplomski rad

Pristupnik: Tino Margan

Naziv diplomskog rada: Generiranje melodija primjenom strojnog učenja

Naziv diplomskog rada na eng. jeziku: Melody generation using machine learning

Sadržaj zadatka: Proučiti načine generiranja melodija, uključujući modelima strojnog učenja. Odabrani barem jedan model kojeg treba opisati i naučiti na vlastitom skupu melodija. Naučeni model treba evaluirati na osnovu srednje ocjene više ispitanika.

Mentor: doc. dr. sc. Miran Pobar

Voditeljica za diplomske radove:
Prof. dr. sc. Ana Meštrović

Zadatak preuzet: 5.6.2022.

(potpis pristupnika)

Fakultet informatike i digitalnih tehnologija
Radmile Matejčić 2, 51000 Rijeka, Hrvatska
http://inf.uniri.hr ured@inf.uniri.hr
Tel: +385 (0)51584700 | Fax: +385 (0)51 584 749

uniri | YUFE

Sadržaj

1. Uvod

Najveću većinu glazbenika zatekli su trenutci u kojima su izgubili inspiraciju prilikom stvaranja pjesme, okolnosti u kojima je bilo nemoguće osmisliti glazbu ili faze stvaralaštva poznate kao kreativna blokada (engl. *writer's block* ili *creator's block*). Kako kreativna blokada zahvaća tekstopisce, isto tako zahvaća i glazbenike koji osmišljavaju melodije, aranžmane, teme, a u tim trenutcima, jedino što glazbeniku preostaje jest tražiti inspiraciju u pjesmama raznih drugih autora ili oslanjati se na pomoć drugih glazbenika s kojima možda svira u grupi i dijeli mišljenja i ukuse.

To je problem kojeg bih pokušao riješiti programom koji može osmisliti jedinstvenu melodiju ni iz čega, program kojeg korisnik može pokrenuti i bez pretjeranog razmišljanja pružiti koliko god je potrebno gotovih, originalnih melodija. Za više personalizirani pristup, taj program trebao bi moći i kao ulazni podatak uzeti ono što je glazbenik do tada osmislio, a na izlazu pružiti potpuno novu i jedinstvenu melodiju koja se nastavlja i temelji na pruženoj ulaznoj melodiji. Na taj način glazbenik može nadomjestiti nedostatak inspiracije melodijama koje navedeni program generira i dovršiti glazbenu misao. Ponekad je samo jedan korak u pravom smjeru dovoljan da se glazbenik u potpunosti riješi blokade i da kreativna inspiracija, u najboljem slučaju, ponovno krene teći. U teoriji zvuči relativno jednostavno, ali načini na koje glazbenici dolaze do melodija nisu jasni ni njima samima.

Objasniti nekome te načine naprosto je nemoguće - ponekad je taj proces jako kompleksan i, pod uvjetom da glazbenik poznaje glazbenu teoriju, može je koristiti da dobije zanimljive, nove i jedinstvene progresije u pjesmama. S druge strane, računalo, osim ako mu glazbenik eksplicitno ne zada određena pravila, ne poznaje glazbenu teoriju, ne zna što zvuči dobro, a što pogrešno, ne zna koji glazbeni žanr pojedina osoba voli ili koja glazba ih nadahnjuje i ne može tek tako osmisliti nešto nalik onome što bi taj glazbenik napisao.

Cilj ovog rada je generirati melodije koristeći strojno učenje. Pod generiranjem melodija podrazumijeva se da računalo temeljem nekog skupa parametara odabere i rasporedi određenu količinu nota u određenu jedinicu vremena. Melodije koje bi računalo generiralo trebale bi biti smislene, iskoristive (imati potencijala prerasti u pravu pjesmu), a u najgorem slučaju trebale bi poslužiti barem kao inspiracija glazbeniku za daljnji rad na melodiji. Kada kažemo da su melodije smislene, možemo zamisliti melodije u funkciji rečenice: ako rečenica ima svoj početak i kraj, ima subjekt, predikat, sve osnovne komponente i **ne** sastoji se od nasumičnih riječi koje su se slučajno našle na istom mjestu - tada je ta rečenica smislena. Na

isti način to možemo reći za melodiju: melodija mora imati svoj početak i kraj, tonovi koji tvore melodiju moraju biti iz istog skupa, povezani u ljestvici i, kao u rečenici, ako se melodijom postavi pitanje, tada melodija mora dati i odgovor. Najbolji primjer smislene melodije koja ima „pitanje-odgovor“ element u sebi je jedna melodija koju svatko zna iz svog djetinjstva, a radi se o Looney Tunes uvodnoj melodiji (1). Prvi dio melodije gleda se kao pitanje (ima neriješen završetak), a za drugi dio melodije u glazbenoj terminologiji moglo bi se reći da se melodija razriješi i da slušatelj ima osjećaj završetka, što se na slušatelja u konačnici prenosi kao osjećaj zadovoljstva.

Melodije bi također trebale biti prilagodljive, primjerice, ako slušamo metal glazbu, ne želimo da nam program generira nekakvu melodiju koja žanrovski nije niti približno tome što mi tražimo. Taj problem pokušat ćemo riješiti u ovom radu korištenjem strojnog učenja. Prvi alat koji ćemo koristiti je Google-ova Magenta, a drugi model bit će osnovan na evolucijskom računarstvu, odnosno genetskom algoritmu. Skup podataka za Magentin model sami ćemo kreirati uz pomoć Lakh skupa podataka. Iz tog skupa izvlačit ćemo samo glavne melodije koje možemo čuti u pjesmama, dakle bez bubnjeva, bez akorda, harmonija, itd., isključivo melodije i to samo iz onih pjesama koje nam se uklapaju u vrstu (žanr) melodije koju želimo generirati. Za početak, proučimo nešto glazbene teorije koja nam je potrebna za daljnji rad.

1.1. Analiza glazbenih djela i glazbene teorije

Proučavanjem glazbenih djela primijetit ćemo da se u najvećoj većini slučajeva ona sastoje od glavnih dijelova na koje se skladatelj uvijek vraća, tj. dijelova koji se ponavljaju više puta kroz pjesmu. Ti ponavljajući dijelovi mogu biti glavna tema, vokalna melodija, dio teksta u refrenu, a nerijetko i cijeli refren. Prva stvar koju pjesma u pravilu mora imati je svoja polazišna točka, mora imati svoju „bazu“, temu, odnosno nešto na što se uvijek može vratiti - inače neće zvučati kao pjesma, već samo kao slijed melodija koje se samo nižu jedna za drugom. Za osobu koja stvara model to znači da model mora imati mogućnost memoriranja prethodnih dijelova melodije koju generira, ali također mora moći napraviti varijacije na zadanu temu da stvori zanimljivost i zadrži slušateljevu pažnju. Repeticija s minimalnim varijacijama je ono što pogotovo glazbenici cijene kod velikih bendova. U mnogo slučajeva kada se ponavlja neki dio pjesme, obično bude jedna nota u sekvenci koja je ili različitog tona ili ekspresijski drukčije odsvirana.

Druga važna komponenta koju pjesma mora zadovoljiti je da zadrži početnu ritmiku i tonalitet kroz cijelo trajanje. Naravno, ima nebrojeno puno pjesama u kojima nailazimo na

modulaciju (promjenu tonaliteta) ili čak i promjenu ključa na određenim dijelovima pjesme kako bi se dinamika zvučne slike podigla na višu razinu. Isto tako nailazimo i na promjenu tempa ili ritmike da različiti dijelovi dođu do većeg izražaja, ali u globalu možemo zaključiti da su ritmika i tonalitet konstantni kroz pjesmu. Model zato mora biti svjestan odnosa između nota, trajanja pojedinih nota kao i pauzi između njih, a poželjno se i držati zadane ljestvice unutar koje će generirati note.

Nešto više glazbene teorije bit će nam potrebno u kasnijim poglavljima gdje će se generirati melodija temeljem danih parametara za koje moramo znati što rade da bi dobili nešto približno onome što želimo. U sljedećem poglavlju osvrnut ćemo se na automatsko generiranje melodija i različite tehnike kojima se ono može izvesti.

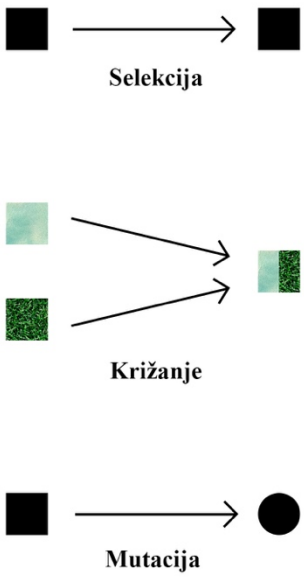
2. Automatsko generiranje melodija

2.1. Generiranje melodija temeljeno na pravilima

Ako računalo dobije uputu da nasumično generira note u proizvoljnom trajanju i ključu, šanse da ćemo dobiti nešto što uopće zvuči nalik melodiji izuzetno su male. Da bi generirali melodiju temeljenu na pravilima, taj skup pravila morao bi biti golem. Morali bi odrediti što je nota, što je pauza, što je takt, što je ljestvica, koje note se nalaze u kojoj ljestvici, koje note međusobno pristaju, a koje ne. Primjerice, svaka ljestvica (C dur, C mol, D dur, D mol, itd.) sastoji se od različitih tonova. Iako postoji samo dvanaest nota, njihov međuodnos je ono što je važno u melodiji, stoga je potrebno računalu zadati pravila. Jedno pravilo moglo bi glasiti: ako se generira melodija u ljestvici C dur, ne smiju se koristiti tonovi C#, D#, F#, G# i A#. Još jedno od pravila moglo bi glasiti: ako je najkraće zadano trajanje note osminka, ne smije se koristiti šesnaestinka niti ijedna druga nota kraćeg trajanja. Na taj način računalo ograničavamo da ne miješa note iz različitih ljestvica i generira melodiju koja objektivno ne zvuči ispravno. Jedan od takvih načina za smanjivanje skupa pravila primijenjen je u (2) gdje se svo generiranje melodije odvija u jednoj ljestvici jednog ključa - C dur. Na taj način ne moramo brinuti o različitim notama iz različitih ljestvica, ne moramo brinuti o tome je li algoritam slučajno prešao u drugu ljestvicu i sl. Time je broj pravila koje moramo odrediti puno manji, ali posljedica tome je da su rezultati bazični te je korisnik i sam ograničen tim pojedinim vrijednostima. Primjerice, ako ne želi više generirati melodije koje su u C dur ljestvici, mora iznova potpuno promijeniti set pravila. Cilj ograničenog programiranja je postepeno smanjivati i ograničavati skup mogućih vrijednosti koje varijable mogu poprimiti sve dok se svakoj varijabli ne dodijeli njena vrijednost. Problem je što korisnik da bi zadao određena pravila programu, mora se dobro razumjeti u glazbenu teoriju i znati što je poželjno, a što nepoželjno kako ipak ne bi dobio melodije koje nemaju pretjeranog smisla.

2.2. Genetski algoritam i generiranje melodija temeljeno na genetskom algoritmu

Općenito govoreći, genetski algoritmi odnose se na skup algoritama inspiriranih osnovnim principima evolucije, a koriste se za probleme optimizacije. Kao što promatramo evoluciju u prirodi gdje svaka jedinka koja se dobro prilagodi okolnim uvjetima ima najveće šanse za preživljavanje, slično genetski algoritam bira najbolje jedinke u danoj populaciji čije se osobine prenose i rekombiniraju u sljedeću generaciju, dok se ne postigne zadovoljavajuće rješenje. Genetski algoritam u svakoj iteraciji ima niz potencijalnih rješenja zvanih jedinke, čija je struktura opisana kodom („kromosomom“) na kojima se primjenjuju genetski operatori. Operatorom križanja, dvije jedinke izmjenjuju gene koji nastaju pri reprodukciji, a operatorom mutacije jedinke mutiraju, tj. slučajno mijenjaju svoj kod, odn. „genetski materijal“. Principom evolucije (selekcije) najsposobnije jedinke opstaju, mijenjaju svojstva, a nedovoljno dobre „izumiru“ (3) (4). U svakom koraku genetski algoritam odabire jedinke iz trenutne populacije koje će biti roditelji i koristi ih za generiranje potomaka za sljedeću generaciju. Generaciju za generacijom, cijela populacija evoluira prema optimalnom rješenju (5).



Genetski algoritam koristi tri vrste pravila u svakom koraku za generiranje nove generacije iz trenutne populacije jedinki: pravilo križanja, pravilo mutacije i selekcijsko pravilo. Pravilo križanja bira dva roditelja koja će stvoriti potomke za sljedeću generaciju, a pravilo mutacije vrši nasumične promjene na pojedinim roditeljima i time oni sami postaju potomci iduće generacije. Mutacija je unarni operator jer djeluje nad jednom jedinkom, a kao rezultat mutacije dobivamo jednu izmijenjenu jedinku (3). Selekcijsko pravilo bira roditelje koji će pridonijeti populaciji sljedeće generacije - od kojih će se generirati potomci. Odabir je uglavnom nasumičan, ali na odabir mogu utjecati neki faktori poput toga koliko je kvalitetna, odnosno dobra ta jedinka, što je definirano funkcijom cilja ili funkcijom dobrote. Svaka jedinka u genetskom algoritmu je potencijalno rješenje problema, a rješenja se u većini primjera gdje je to primjenjivo prikazuju binarnim prikazom. Broj potencijalnih rješenja predstavlja onoliko binarnih vektora kolika je veličina populacije, a optimizacija se događa dok god određeni uvjet (najčešće predodređeni broj iteracija) nije zadovoljen (3).

U pristupu generiranja melodija temeljenom na evolucijskom računarstvu iterativnim postupkom se kroz niz generacija operacijama križanja i mutacije od početnih nasumičnih melodija dolazi do skupa melodija koje bolje zadovoljavaju neku kriterijsku funkciju, ili funkciju cilja. U ovom slučaju i dalje može postojati skup pravila za generiranje inicijalnih melodija ili za definiciju operatora mutacije i križanja, ali skup pravila više ne mora jasno uređivati sve aspekte generiranja melodija do najsitnijeg detalja. Prema (6) postoje tri glavne varijante za generiranje melodija pomoću genetskog algoritma, a to su korištenjem „objektivne“ funkcije cilja, uključivanje mišljenja korisnika umjesto funkcije cilja i korištenjem neuronske mreže kao funkcije cilja. Uloga funkcije cilja ili funkcije dobrote je procijeniti koliko je određena „jedinka“ dobra za dani zadatak, odnosno, u slučaju melodija, koliko je melodija dobra. Cilj genetskog algoritma je maksimizirati funkciju cilja, pa je odabir te funkcije ključan za kvalitetu rezultata. Definicija funkcije cilja za procjenu dobrote neke melodije nije jednostavna jer ne postoje objektivna i jednoznačna pravila za evaluaciju melodije i glazbe općenito. S obzirom da nije posve jasno prema kojim kriterijima bi se objektivno ocijenila određena generirana melodija, ovakva definicija funkcije cilja za generiranje glazbe nije često korištena.

Ulogu funkcije cilja može preuzeti i korisnik koji ocjenjuje melodije, no i taj pristup ima nekoliko problema, a glavni su da gubimo objektivnost i gubimo dosljednost. U YouTube videu u kojem korisnik ocjenjuje generirane melodije (7) možemo vidjeti da u jednoj populaciji melodiju ocijenio ocjenom 2, a nakon manje od minute u drugoj populaciji od note do note identičnu melodiju ocijeni ocjenom 4. Kao razlog tome u istraživanju navodi se da na naše trenutno mišljenje utječu melodije koje smo čuli netom prije, utječe ako iskusimo promjenu volje pa čak i ako nam je jednostavno dosadno. Osim toga, učinkovitost je velik problem - u takvim zadacima čovjek se ne može mjeriti s računalom isključivo u pogledu brzine. Postoji još jedan problem, za čiju ilustraciju možemo zamisliti sljedeći scenarij: zadamo računalu da nam generira šest melodija koje moramo ocijeniti ocjenama od 1 do 5. Dvije najbolje melodije prenose se u sljedeću populaciju i na njima se vrše funkcije mutacije i križanja, dok se ostale melodije odbacuju. Prva melodija koju računalo generira bude izvrsna i ocijenimo je ocjenom 5. Nakon te melodije računalo nam generira još preostalih nekoliko melodija za koje se ispostavi da su bolje od one prve melodije koju smo čuli, a mi preostalim melodijama naprosto nemamo za dati veću ocjenu od veća dane petice. Ponekad se može dogoditi i suprotno - možemo prvoj melodiji dati ocjenu 2 ili 1, a da se naposljetku ispostavi da će u toj populaciji upravo ta melodija biti najkvalitetnija.

Korištenje neuronske mreže kao funkcije cilja potencijalno ima najviše smisla jer uz dovoljan broj označenih primjera „dobrih“ i „loših“ melodija neuronsku mrežu možemo naučiti da konzistentno ocjenuje što je „dobra“ melodija, a što je loša melodija. Na taj način dobiva se ponovljivost i brzina objektivne funkcije, uz ugrađeno subjektivno mišljenje ljudskih slušaoca koje je teško eksplicitno izraziti skupom pravila. Dodatno, kako bi naučili neuronsku mrežu što su to loše melodije, možemo uzeti melodije koje je sam model generirao za koje smatramo da nisu dobre i iznova ih koristiti u treniranju modela.

2.3. Generiranje melodija temeljeno na strojnom učenju

U ovom radu osvrnut ću se na dobivanje melodija oslanjajući se na koncept strojnog učenja. Strojno učenje uglavnom se koristi za probleme gdje rukujemo s velikim brojem podataka, gdje nismo sigurni kako bi tradicionalnim pristupom riješili traženi problem, problem klasifikacije, generalizacije, predviđanja, itd.

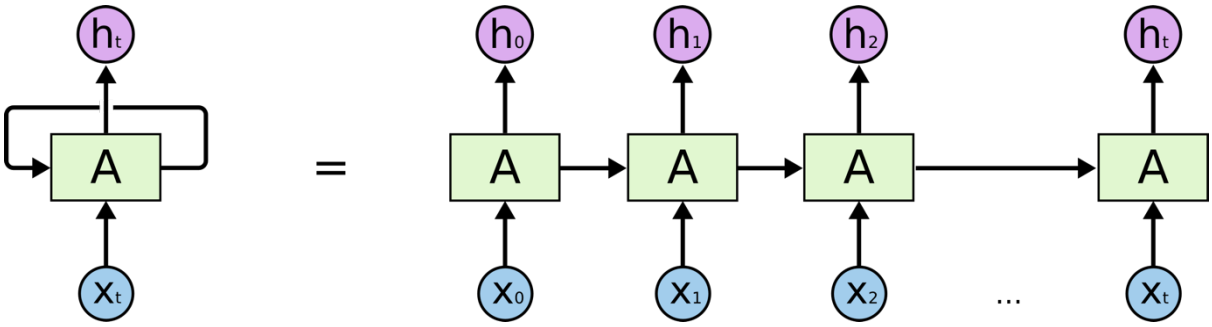
Strojno učenje je znanost u grani programiranja u kojoj računala uče zakonitosti iz podataka bez da ih se eksplicitno programira (8). Dobar primjer primjene strojnog učenja su pomoćnici s kojima možemo komunicirati na našim pametnim uređajima kao što su Siri, Google Assistant ili Alexa. Sposobni su raspoznati izgovorene naredbe i na osnovu njih izvršiti određene radnje. Pomoćnici su svakim danom sve bolji - bolji u raspoznavanju riječi, bolji u pružanju pomoći, čak i raspoznaju različite glasove na istom uređaju i pružaju personalizirane rezultate ovisno o osobi koja se obraća asistentu.

S tradicionalnim pristupom programiranju, takvi ne-trivijalni problemi poput raspoznavanja govora koje podrazumijeva prevođenje zvučnog signala u sekvencu koju čine izgovorene riječi na način da računalo to razumije bili bi praktički neizvedivi za riješiti. Iziskivali bi veliki skup pravila koja bi programer trebao implementirati u program i bili bi nemogući za održavanje jer bi se vremenom konstantno morao unositi novi skup pravila koja se mijenjaju iz dana u dan.

U radu govorimo samo o nadziranom strojnom učenju gdje ćemo se osvrnuti na primjer Melody RNN mreže unutar okoline Magenta, te ga usporediti s postupkom generiranja melodija korištenjem genetskog algoritma kojeg ćemo sami kreirati. Algoritam nadziranog strojnog učenja stvara pravila na osnovu skupa za treniranje koji se sastoji od parova uzoraka i željenog izlaza iz sustava. Postoji također i nenadzirano strojno učenje koje se od nadziranog razlikuje po tome što podaci na kojima se model trenira nemaju tu ključnu varijablu koja modelu govori koji je ustvari željeni izlaz, tj. model nastoji učiti bez učitelja. Problemi koji se

obično rješavaju nenadziranim strojnim učenjem su problemi grupiranja ili vizualizacije (9). Primjere modela nadziranog strojnog učenja i genetskog algoritma vidjet ćemo u kasnijim poglavljima gdje koristimo te modele za generiranje glazbenih melodija. Koristeći genetski algoritam, korisnik je taj koji modelu govori što je dobro, a što nije dobro i na taj način kroz nekoliko iteracija, u idealnoj situaciji, izlazni proizvod tog modela postaje sve bolji i bolji.

Suprotno od genetskog algoritma, za učenje Melody RNN modela uopće nije potreban korisnik koji evaluira generirane melodije. Melody RNN je nadzirano naučen model kod kojeg je problem generiranja melodije sveden na problem predviđanja note koja slijedi nakon sekvence prethodnih nota. Za učenje modela nije potreban korisnik koji kaže što je dobro, a što nije, nego velik broj melodija na kojima model uči predviđanje sljedeće note. Model na osnovu podataka uči distribuciju vjerojatnosti idućih nota, a nakon što Melody RNN generira jednu novu notu, ta nota u sljedećem koraku također ulazi u model i postaje dio melodije na kojem se vjerojatnost sljedeće note opet uči. Na taj način obuhvaćamo svaki novo-generirani korak i iznova ga dodajemo kao ulazni podatak kako bi dobili cijelu, smislenu melodiju (10).



Slika 1 - prikaz povratnih veza u neuronskoj mreži kroz vrijeme (10)

Osim Melody RNN koje se temelje na LSTM mreži, u literaturi možemo pronaći čak i veoma uspješne pokušaje generiranja melodija, pa čak i cijelih glazbenih kompozicija korištenjem generativnih suparničkih mreža (GAN, engl. ***G**enerative **A**dversarial **N**etworks*) i varijacijskih autoenkodera (VAE, engl. ***V**ariational **A**uto**e**ncoders*). Generativno modeliranje je nenadzirana vrsta učenja u strojnom učenju koja uključuje automatsko raspoznavanje i učenje uzoraka u ulaznim podacima. Takav model toliko dobro prepoznaje uzorke da se izlaz iz modela bez ikakvih poteškoća može koristiti za generiranje novih podataka na isti način kao da su iz originalnog skupa podataka. Generativne suparničke mreže prvi put se spominju u radu Iana Goodfellowa i drugih, naziva „Generative Adversarial Networks“ (11), a kako i samo ime govori, sastoje se od dva pod-modela od kojih jedan model generira podatke, a drugi nastoji klasificirati jesu li ti podaci stvarni (već postoje) ili su (12)nepostojeći, odnosno generirani. Ta dva modela nazivaju se generator, odnosno diskriminator. Ta dva modela treniraju se zajedno,

paralelno, sve dok diskriminator pogrešno ne klasificira oko 50% izlaza iz generatora - to je pokazatelj da generator generira doista vjerodostojne podatke. Generativne suparničke mreže pokazale su se izvrsne u problemima generiranja realističnih primjera prebacivanja fotografija iz zimskog godišnjeg doba u ljetno ili generiranja fotografije noćne scene iz fotografija koja je fotografirana po danu. Također, generiranje foto-realističnih objekata, scena, panorama pa čak i osoba do te mjere da niti ljudi ne mogu raspoznati je li fotografija stvarna ili generirana (13).

Umjesto generiranja melodija na temelju nota, u istraživanju (14) koriste se cijeli taktovi kako bi model točnije prepoznao, naučio, a zatim i iskoristio kompleksne i raznolike značajke glazbe koje se možda ne prenose dovoljno dobro na bazi nota, već na bazi nekoliko nota, odnosno taktova. U modelu su čak dva diskriminatora, jedan je osnovan na konvolucijskoj neuronskoj mreži (CNN), a drugi na neuronskoj mreži s povratnim vezama (RNN). Diskriminator osnovan na RNN-u promatra karakteristike melodije i kako se ona mijenja u vremenu, dok diskriminator osnovan na CNN-u promatra cjelokupnu strukturu melodije. Usporedba generirane melodije sa stvarnom postojećom melodijom vrši se korištenjem TF-IDF algoritma (engl. ***T**erm **F**requency - **I**nverse **D**ocument **F**requency*). TF-IDF algoritam najčešće je korišten u radu s tekstualnim oblicima, a dijeli se na dva dijela. Prvi dio je učestalost izraza (engl. *term frequency*) i tu se doslovno mjeri učestalost izraza (riječi, fraze) u danom dokumentu. Učestalost se može mjeriti u apsolutnom smislu (konkretan broj pojavljivanja izraza u danom dokumentu) ili u relativnom smislu (broj pojavljivanja izraza u dokumentu, ali uzevši u obzir ukupnu količinu riječi tog dokumenta). IDF dio algoritma uspoređuje učestalost izraza s korpusom teksta. Razlog zbog kojeg je IDF algoritam potreban je taj što imamo riječi u hrvatskom jeziku poput veznika „a“, „i“, „ali“, prijedloga „na“, „u“ koje su jako učestale i time pridobivaju veću težinu negoli bi trebali. IDF algoritmom možemo minimizirati težinu učestalih riječi tako da one koje su manje učestale dobiju veću važnost (15).

Isti taj princip možemo primijeniti i na melodije. Učestalost izraza promatra koliko često se ponavlja određena nota ili, još bolje, skupina nota koja možda čini neku temu. IDF algoritam provjerava tu učestalost u cjelokupnoj melodiji i raspoređuje težine notama tako da one koje su važnije za melodiju dobiju prednost nad onima koje su manje važne. Sveukupno, na dvadeset treniranih melodija, ukupna razlika između pravih podataka i melodija koje je osmislio čovjek i melodija koje su generirane naprednom generativnom suparničkom mrežom iznosi svega 8%, što znači da je metoda izuzetno efektna (14).

3. Implementacija modela za generiranje melodija

U ovom radu usporedit će se tri modela za generiranje melodija - dva modela temeljena su na strojnom učenju iz biblioteke Magenta te jedan model na temelju genetskog algoritma. Od dva modela iz biblioteke Magenta jedan je javno dostupan, nepromijenjen model direktno s Magentine web stranice, dok je drugi treniran na vlastitom skupu podataka.

3.1. Magenta

TensorFlow je platforma otvorenog koda koja u sebi ima alate, resurse i biblioteke za razvoj aplikacija koje koriste strojno učenje. TensorFlow pronalazi svoju primjenu u medicini, školstvu, društvenim mrežama i mnogim drugim područjima, a Magenta je biblioteka temeljena na TensorFlow-u koja se koristi za različite eksperimente s glazbom.

Magenta je projekt koji istražuje načine na koje strojno učenje može doprinijeti stvaranju umjetnosti, bilo u vizualnom ili u glazbenom obliku. Također je otvorenog koda, a koristi se kako bi umjetnicima proširila vidike i pružila nove mogućnosti pri kreiranju svojih djela - od web sučelja u kojem aplikacija automatski napravi ilustraciju koju korisnik započne običnom crtom, krugom ili oblikom pa sve do *standalone* aplikacija koje mogu simulirati cijeli bend i zasvirati uz korisnikovu melodiju. Magentu naravno mogu koristiti i oni koji nemaju umjetničku crtu i uz njenu pomoć mogu kreirati prateći sadržaj za bilo što čime se bave, primjerice generiranje pozadinske glazbe za YouTube video, generiranje ilustracija ili fotografija koje prate glazbu, virtualni bubnjar za korisnika koji svira gitaru, itd. To ustvari i jest glavna nit vodilja Magente, osmisliti način da kreativnost bude dostupna svima i da je jednostavno započeti s njenim radom. U idućem poglavlju bit će prikazan način kako trenirati vlastiti model strojnog učenja putem Magente i to u svega nekoliko linija u naredbenom retku. Magenta se može koristiti kao Python TensorFlow biblioteka ili kao Magenta.js (TensorFlow.js) biblioteka koja se može pokrenuti u internet pregledniku.

3.2. Instalacija i korištenje Magente

Magentu je moguće koristiti na lokalnom računalu ili u oblaku, npr. na besplatnom servisu u oblaku Google Colab temeljeno na Jupyter dokumentima, bez potrebe za ikakvom instalacijom lokalno na računalu.

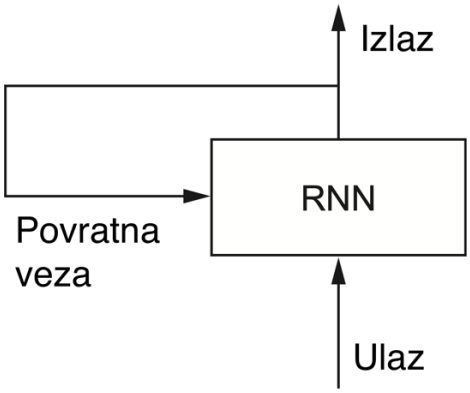
Iako je ovaj način rada jednostavniji za započeti, u projektu radimo s datotekama koje bi trebalo prenositi na server i preuzimati sa servera, kretati se po mapama, itd. To je jednostavnije za učiniti na računalu i koristiti lokalne datoteke bez ikakvog prenošenja. Neke

stvari na koje treba obratiti pozornost su verzija Pythona koja se koristi, kao i verzija TensorFlow-a. Kombinacija verzija koja uspijeva na jednom računalu ne garantira uspjeh na drugom zbog velikog broja međuovisnosti raznih paketa tako da valja probati sve kombinacije. U radu je korišteno računalo s Windows 10 operativnim sustavom, Python 3.6.0 i TensorFlow 2.6.2. Nakon instalacije Magente, potrebno je još i preuzeti datoteke vezane za model koji ćemo koristiti, u ovom radu koristit ćemo `melody_rnn` i nakon toga smo spremni za početak rada. Kao što samo ime govori, model je specijaliziran za rad s glazbenim melodijama i radi na principu neuronske mreže s povratnim vezama.

Pored Melody RNN modela, Magenta raspolaže brojnim drugim modelima. *Improv_rnn* radi na sličnom principu kao i *melody_rnn*, ali se oslanja na progresiju akorda u podlozi povrh koje, kako i samo ime modela kaže, računalo generira improvizacijsku melodiju. Akord tvori nekoliko nota (dvije ili više) koje su odsvirane odjednom, a progresija akorda znači slijed u kojem se akordi sviraju. Primjer jednog slijeda akorda bio bi da prvo ide akord C dur, zatim D dur, G dur, A dur i povratkom na C dur zatvaramo jedan ciklus progresije. Još jedan model vrijedan spomena je *polyphony_rnn* koji, za razliku od *melody_rnn* modela ima sposobnost generiranja akorda, dakle note se mogu preklapati, melodije ne moraju biti linearne i otvaraju se mogućnosti za kompleksne skladbe. Ovaj konkretan model treniran je na djelima Johanna Sebastiana Bacha, što znači da, koju god notu, akord ili melodiju pružimo modelu na ulazu, dobit ćemo kompleksnu baroknu skladbu koja doista zvuči kao da ju je skladao sam Bach.

3.3. Melody RNN

Melody RNN je model koji generira melodije koristeći LSTM (engl. ***L**ong **S**hort-**T**erm **M**emory*) što je vrsta neuronske mreže s povratnim vezama razvijena 1997. godine. Suprotno od mreža koje nemaju memoriju, tj. mreža u kojima svaki se ulazni podatak obrađuje zasebno, bez ikakve međusobne poveznice, neuronske mreže s povratnim vezama obrađuju podatke u sekvenci, kontekstualno, memorirajući ono što je prethodilo kao ulazni podatak (16). Takva vrsta mreže sposobna je naučiti redoslijed i međusobnu ovisnost pojedinih elemenata u problemima predviđanja (17). Prema (18) neuronske mreže s povratnim vezama sadrže neurone koji kao ulazne podatke i kao aktivacijsku funkciju koriste izlazne podatke iz prethodnog vremenskog ciklusa.



Slika 2 - grafički prikaz neuronske mreže s povratnom vezom (petljom) (16)

Ti podaci spremljeni su u neuronskim vezama koje pohranjuju dugoročne kontekstualne informacije. Na taj način neuronske mreže s povratnim vezama iskorištavaju te podatke za bolje generiranje glazbenih melodija kroz određeno vrijeme, a upravo to je ono što je u glazbi najvažnije - međusobna ovisnost nota koja se razvija kroz ponešto dulji period vremena. U radu s neuronskim mrežama najčešće se povlače paralele s radom ljudskog mozga, pa tako i u (12) gdje se govori o tome kako mi kao ljudi ne započinjemo svoje razmišljanje iznova svaki put kada dobijemo novu informaciju. Naš mozak sposoban je tu informaciju obraditi u odnosu na kontekst onoga što znamo otprije. Upravo na taj način rade i neuronske mreže s povratnim vezama.

LSTM mreže započinju kao jednostavne mreže s povratnim vezama - pojednostavljeno, `for` petlja koja u svom izračunu koristi podatke koji su se obrađivali u prethodnoj iteraciji petlje (16).

```
state_t = 0
for input_t in input_sequence:
      output_t = f(input_t, state_t)
      state_t = output_t
```

pri čemu je `state_t` varijabla stanja mreže koja ovisi o svim prethodnim ulazima do vremenskog koraka t, `input_t` je ulaz u tom vremenskom koraku, funkcija `f` je transformacijska funkcija koja računa trenutni izlaz na osnovu trenutnog stanja `state_t` i ulaza `input_t`. Na kraju varijabla `state_t` poprima vrijednost izlazne varijable i time služi kao varijabla stanja za sljedeću iteraciju (iteracije se vrše na elementima željene sekvence) (16).

Kod LSTM mreža postoji dodatna varijabla prijenosa (carry) koja se proteže cijelom mrežom i prenosi stanje ćelija u danom vremenu. Ta varijabla može prolazi kroz više iteracija neuronske mreže bez previše interakcije, čime se mogu prenositi ključne informacije koje ostaju nepromijenjene. Mreža ima mogućnost regulirati informacije koje dolaze do navedene varijable putem sigmoidne funkcije koja na izlazu brojevima između 0 i 1 opisuje koliko koje informacije treba propustiti do te varijable (broj 0 znači da se ne propušta ništa, dok broj 1 znači da se propušta sve). Ta varijabla ustvari odlučuje koje informacije neće utjecati na model, a koje će zapamtiti i koje su važne (12). Najvažnija stvar, dakle, koju LSTM mreža donosi je mogućnost ubacivanja podataka u mrežu, i to podataka koji su se dogodili u nekom prošlom vremenu, što znači da nema podataka koji vremenom izgube na važnosti ili „izblijede“ (16).

3.3.1. Opis modela

Model na kojem se temelji Melody RNN je osnovna dvoslojna LSTM mreža sa 64 ili 128 jedinica po sloju, a ima četiri konfiguracije koje koristi za generiranje melodija. Od jednostavnijih prema kompleksnijim zovu se *Basic*, *Mono*, *Lookback* i *Attention*.

Basic je, kako mu i ime glasi, najosnovniji algoritam za generiranje melodija, koristi *one-hot* način kodiranja nota (vektori s nulama na svim pozicijama osim jedne jedinice na onoj poziciji na kojoj se nalazi nota), a sve note su transponirane unutar raspona od tri oktave MIDI nota. *Mono* algoritam je u svemu potpuno jednak osnovnom algoritmu, ali umjesto raspona od tri oktave ima puni raspon od 128 nota, tj. nešto više od 10 oktava (19).

U *Basic* i *Mono* algoritmima jedino što mreža ima na ulazu je navedeni *one-hot* kodirani vektor prethodne note - i to je sve, jedino što taj model „zna“ jest koja je visina prethodne note i koje je njeno trajanje. Primjer kodiranja *one-hot* vektora za notu C4 bio bi [0,0,0,...0,1,0,...,0,0,0], pri čemu se koristi vektor veličine 128 znamenki s jednom jedinicom i ostalim nulama. U MIDI zapisu gdje imamo raspon od 128 nota, note mogu poprimiti vrijednosti od 0 do 127, što znači da će C4 kojem je dodijeljen broj 60 točno na toj poziciji u vektoru imati jedinicu. Brojevi od 0 do 20 odgovaraju frekvencijama od 8 Hz do 26 Hz što je dobrim dijelim ispod zvučnog spektra prosječne osobe.

Za generiranje sljedeće note, algoritam ima tri opcije: 1. zaustaviti notu koja se u tom trenutku reproducira, 2. nastaviti reproducirati što god da se u tom trenutku reproducira (ako je tišina - ostaje tišina) i 3. zaustaviti notu koja se u tom trenutku reproducira i pokrenuti drugu notu u određenom tonu. Generiranje melodija u Melody RCNN zasnovano je na šesnaestinkama, dakle ako želimo da nota traje dulje od šesnaestinke postoji poseban zapis koji

treba koristiti da bi se to napravilo. Pri unosu parametara u Magenti, kako je već rečeno, brojevi od 0 do 127 označavaju koja nota se reproducira, međutim broj -2 odgovara prvoj opciji - zaustavljanju note koja se u tom trenutku reproducira, dok broj -1 odgovara drugoj opciji - nastaviti reproduciranje onoga čega se reproducira (ako je pauza, tada se nastavlja pauza). Primjerice, ako želimo jednu šesnaestinku u ključu C4 napisat ćemo „[60]“. Ako želimo dvije šesnaestinke za redom, napisat ćemo „[60, 60]“. Ako želimo jednu osminku (trajanja dvije šesnaestinke) u istom ključu C4, napisat ćemo „[60, -1]“, a ako želimo jednu šesnaestinku note C4 i jednu šesnaestinku pauze, napisat ćemo „[60, -2]“. Postavlja se pitanje, kako ovaj pseudo-zapis pretvoriti u *one-hot* vektor koji Magenta inače koristi? Prema (20) kako bi se znalo nastavlja li se nota ili se na nekom mjestu nalazi pauza, potrebno je u *one-hot* vektor uvrstiti još dvije varijable koje govore upravo to, nastavlja li se prethodna nota (ili pauza) ili se zaustavlja (i nastaje pauza) (engl. *hold* i *rest*). Dakle, umjesto vektora sa 128 pozicija, vektor ima 130 pozicija u kojima dodatne dvije pozicije to i kodiraju. Za stvaranje kompleksnih melodija koje imaju uzorke koji se protežu kroz nekoliko taktova, nailazimo na *Lookback* i *Attention* konfiguracije.

3.3.2. Lookback i Attention

Lookback i Attention RNN više ne prepoznaju samo visinu i trajanje prethodne note, već visinu i trajanje nota u prethodna dva takta. Ako je melodija repetitivne prirode to znači da je veća vjerojatnost da će se ista melodija ponoviti nakon jednog ili nakon dva takta, ovisno o prirodi i samom trajanju ciklusa melodije. Lookback RNN to radi tako da u ulazni vektor dodaje prilagođene ulaze kojima model jednostavnije prepoznaje uzorke u melodiji koji se, umjesto samo u jednoj prethodnoj noti, javljaju u prethodna dva takta. Prilagođeni ulazi također olakšavaju modelu prepoznavanje uzoraka u vremenskom smislu (u kojem dijelu takta se nalazi nota). Uz prilagođene ulaze, također nalazimo i prilagođene oznake kojima model može ponoviti čitavu sekvencu nota u melodiji bez da se išta čuva u netom spomenutoj varijabli za stanje ćelija. Uz to, svaka nota se provjerava je li generirana na istom mjestu kao i pred jedan takt, odnosno pred dva takta i na taj način model bolje zna je li u stanju repeticije ili u stanju generiranja nečeg novog. To se radi pomoću prilagođenih oznaka koje se dodjeljuju notama koje se ponavljaju u odnosu na prethodni takt, odnosno na notu pred dva takta (*repeat-1-bar-ago* i *repeat-2-bars-ago*).

Kada želimo prepoznavanje uzoraka u melodiji koje seže dulje od jednog ili dva takta, tada ćemo koristiti Attention RNN koji pri generiranju note promatra prethodnih n nota koristeći matematičke formule:

$$u_i^t = v^T \tanh\left(W_1' h_i + W_2' c_t\right)$$

$$a_i^t = softmax(u_i^t)$$

$$h_t' = \sum_{i=t-n}^{t-1} a_i^t h_i$$

U formulama, vektor v i matrice W_1' i W_2' su parametri koje mreža uči, to su uglavnom težine koje su dodijeljene svakoj vezi (jačina veza između neurona u neuronskoj mreži - realni broj s kojim se množi izlazni signal neurona i određuje utjecaj veze na mrežu) i pristranosti (uglavnom realna konstanta koja se zbraja umnošku ulaznog signala s težinom) (21). Izlazi iz neuronske mreže za prethodnih n nota prikazani su s h_i (pri čemu je $i = t - n, \dots, t - 1$), dok je vektor c_t stanje ćelije trenutne note. Te vrijednosti unose se u prvu formulu i dobiva se vektorska vrijednost za svaku od prethodnih n nota koje modelu opisuju koliko „pažnje“ (engl. attention) bi svaka od tih nota trebala dobiti. Softmax sloj služi za normalizaciju vrijednosti koje se pospremaju u *attention mask* a_i^t. U posljednjoj formuli te vrijednosti množimo sa svakom vrijednošću s izlaza neuronske mreže za prethodnih n koraka i međusobno ih zbrajamo da bi dobili vektor h_t' koji kombinira izlaze prethodnih n koraka, ali uz uvjet da svakom izlazu pridaje njegov individualni dio pažnje, kako je i izračunato (22).

3.4. Generiranje melodija korištenjem Melody RNN

Za generiranje melodija korištenjem Melody RNN koristi se naredba u naredbenom retku s ključnim riječima „*melody_rnn_generate*“, a tipičan kod s potrebnim parametrima za generiranje melodija izgleda ovako:

```
melody_rnn_generate
        --config=attention_rnn
        --bundle_file=C:/Users/magenta/attention_rnn.mag
        --output_dir=C:/Users/magenta/melody_rnn/generated
        --num_outputs=10
        --num_steps=128
        --primer_melody="[60]"
```

Prvi parametar koji moramo definirati je konfiguracija koju želimo koristiti (*Basic*, *Mono*, *Lookback* ili *Attention*). *Bundle* datoteka u sebi sadrži razne metapodatke o modelu, funkciju cilja i sve što je bilo korišteno u treniranju tog modela. Potrebno je odrediti mjesto na

računalu gdje će se pohraniti izlazne MIDI datoteke, koliko datoteka (melodija) će se generirati i trajanje melodija u šesnaestinkama (16 koraka = 1 takt). Primjerice ako generiramo 128 koraka, to znači da ćemo dobiti 8 taktova melodije, za 256 koraka, dobit ćemo 16 taktova melodije, itd. Svaka MIDI datoteka koju Magenta generira bit će, naravno, jedinstvena melodija i neće imati međusobne poveznice s drugim generiranim melodijama.

Zadnji parametar može glasiti `--primer_melody` ili `--primer_midi`. Ako iskoristimo primer_melody, znači da će nam Magenta generirati melodiju od samog početka, dok primer_midi zahtijeva od nas da na ulazu pružimo MIDI datoteku koja sadrži melodiju na koju se Magenta može nadovezati. Dakle izlazna datoteka u tom slučaju bit će cijela ulazna melodija plus generirana melodija koja se nastavlja na ulaznu melodiju. Potrebno je naglasiti da niti u jednom pokušaju nastavljanja melodije ona nije započinjala u dobroj ritmici, tj. uvijek bi započela jedan korak kasnije od trenutka kada se od melodije očekuje da započne. Taj problem pokušao sam riješiti na način da na ulaznoj melodiji obrišem posljednju notu da se možda izbjegne slučajno preklapanje, ali bilo je bez uspjeha. Kod primer_melody parametri koje mi možemo odrediti su intonaciju (tonalitet) s kojom će melodija započeti, odnosno početni ton (*root*) ili početnih nekoliko tonova melodije.

Primjerice `--primer_melody="[60]"` označava da melodija započinje tonom C, točnije tonom C4, dok `--primer_melody="[60, -2, 60, -2, 67, -2, 67, -2]"` modelu govori prve četiri note u dječjoj pjesmi „Blistaj, Blistaj, Zvijezdo Mala“. Kako smo prethodno već utvrdili, broj 60 označava ton C4, broj 67 označava ton G, a brojevi -2 modelu govore da na tim pozicijama ne generira nikakvu notu, tj. da ostavi pauzu (19).

4. Treniranje vlastitog modela koristeći Magentu

Standardni Melody RNN algoritam treniran je na broju melodija koji prelazi desetke, ako ne i stotine tisuća, u njega ulaze svi svjetski žanrovi i ponekad je teško očekivati da će melodija biti baš onakva kakvom ju želimo. Jedan od načina da povećamo šansu za dobivanjem melodije kakvu tražimo je da sami treniramo model iste strukture na skupu podataka koji bolje odgovara tipu melodija koje želimo. Na taj način možemo trenirati model isključivo na glazbi, pjesmama i melodijama koje nas nadahnjuju i konačni proizvod bi, u teoriji, trebao biti puno sličniji toj vrsti melodija.

4.1. Skup podataka

Skup podataka koji je korišten za treniranje vlastitog modela uz pomoć Magente zove se „Lakh MIDI Dataset“ (23). To je skup podataka (24) koji broji čak 176.581 MIDI datoteku, a unutar tog skupa ima brojne pročišćene verzije u kojima su uklonjeni duplikati, MIDI datoteke s greškom i datoteke koje bi mogle prouzročiti probleme u pripremi skupa podataka. Umjesto cijelog Lakh Dataseta, odabrao sam preuzeti „Clean MIDI“ podskup u kojem se nalazi nešto više od dvije tisuće izvođača s ukupno više od devetnaest tisuća glazbenih djela. Svako od tih djela činila je pripadajuća MIDI datoteka koja je u sebi sadržala sve instrumente koji se izvode u tom djelu, dakle svi instrumenti, sve dionice, bubnjevi, gitare, vokali, trube, violine, sve je raspisano u jednoj MIDI datoteci.

MIDI (***M**usical **I**nstrument **D**igital **I**nterface*), odnosno digitalno sučelje za glazbeni instrument, industrijski je standard koji opisuje protokol za komunikaciju, digitalno sučelje i sve što je potrebno za spajanje elektroničkih glazbenih instrumenata i kontrolera s računalom za snimanje ili reprodukciju glazbe (25). MIDI ne prenosi značajke pojedinog instrumenta, ne zna radi li se o klaviru, bubnju, trombonu ili flauti - MIDI isključivo prenosi visinu tona, trajanje tona i glasnoću tona (jačinu udarca). Nakon što se te informacije zapišu u računalni program koji obrađuje MIDI datoteke (poput Pro Tools, Logic Pro, Ableton Live, itd.), na računalu je da simulira instrument po želji korisnika, a MIDI zapis služit će isključivo kao okidač za pojedine note ili elemente odabranog instrumenta.

4.1.1. Priprema skupa podataka

Kako model treba učiti na pojedinim melodijama, moramo znati kako dobiti individualne instrumente iz svake od preuzetih MIDI pjesama, a za to ih je potrebno raščlaniti na individualne komponente, odnosno na pojedine instrumente. Python biblioteka

„pretty_midi“ pruža nam upravo tu mogućnost podjele MIDI datoteke na njene sastavne dijelove kako bi se mogla dobiti samo melodija na kojoj će se trenirati model. Biblioteka „pretty_midi“ oslanja se na to da se različiti instrumenti unutar MIDI datoteka predstavljaju različitim identifikacijskim ključevima (naravno, to je isključivo na osnovi preporuke - nitko ne može zabraniti da se klavirska dionica reproducira na gitari). Ti identifikacijski ključevi su standardizirani i uniformirani, primjerice broj 1 je veliki akustični klavir, broj 33 je akustična bas gitara, broj 41 je violina, itd. (26). Pomoću tih ključeva „pretty_midi“ može izvesti, odnosno izvući pojedine instrumente i jasno ih imenovati za daljnju manipulaciju. Sljedeći kod prikazuje primjer kako se iz jedne MIDI datoteke, koja u sebi sadrži sve instrumente, izdvajaju pojedini instrumenti i njihove dionice spremaju u zasebne, također MIDI datoteke:

```
for i in range(0, len(midi_data.instruments)):
    instrument = midi_data.instruments[i]
    program_num = midi_data.instruments[i].program
    ref = pretty_midi.PrettyMIDI()
    ref.instruments.append(instrument)
    ref.write(str(title) +'_ins_' + str(program_num) + '.mid')
```

Ono što sam tražio u MIDI datotekama su melodije koje predstavljaju pjesmu - to nisu nužno vokalne melodije jer u većini slučajeva vokalne melodije ustvari nisu glavne u definiranju pjesme, već su glavne melodije one tematske prirode koje su odsvirane na nekom od instrumenata. Primjerice, koliko god je vokalna melodija važna i upečatljiva u pjesmi „We Will Rock You“ grupe Queen, i dalje su ta ikonična dva udarca nogom i jedan pljesak glavni nositelji cijele pjesme. Neki od primjera upečatljivih melodija su „The Final Countdown“ grupe Europe, „Crazy Train“ Ozzy Osbournea, „Sweet Child O' Mine“ grupe Guns N' Roses.

4.1.2. TFRecord

Nakon što smo podijelili MIDI datoteke na njihove sastavne dijelove pomoću biblioteke „pretty_midi“, kako od svake pjesme dobijemo između 5 i 10 datoteka zasebnih instrumenata, potrebno je pronaći onu datoteku koja sadrži traženu melodiju i nju iskoristiti u treniranju Magentinog modela. Takvih datoteka koje su korištene u treniranju bilo je točno 50, što nije veliki skup podataka, ali je jako specifičan i sve melodije koje su se u njemu našle imaju nešto zajedničko. Iako MIDI datoteka s traženom melodijom ima svega nekoliko kilobajta, ona je i dalje prevelika i kompleksna za rad te se sve MIDI datoteke koje želimo koristiti za treniranje modela kombiniraju u jednu jedinstvenu datoteku s nastavkom

„.tfrecord“. TFRecord datoteke služe kao međuspremnik protokola za serijalizaciju strukturiranih podataka (nešto poput XML datotečnog formata ali manje, brže za procesiranje i jednostavnije za korištenje). Nakon definiranja načina strukturiranja podataka, posebno generirani izvorni kod može se koristiti za jednostavno pisanje i čitanje tih strukturiranih podataka čak i korištenjem različitih programskih jezika (27). Magenta je za tu konverziju MIDI datoteka u TFRecord strukturu predvidjela jednostavnu komandu `convert_dir_to_note_sequences` čime dobivamo željenu datoteku s kojom možemo započeti treniranje modela.

4.2. Proces treniranja

Kako je već rečeno, sve u radu s Magentom iziskuje svega nekoliko linija teksta u naredbenom retku pa tako i cjelokupni proces treniranja modela. Osim TFRecord strukture, možemo napraviti i primjere sekvenci melodija koje se izvlače iz same strukture i ubacuju u model, jedna tijekom procesa treniranja i druga tijekom procesa evaluacije. Omjer melodija koji će se koristiti za evaluaciju određuje se atributom `--eval_ratio` gdje se s omjerom od 0.2 20% izvučenih melodija sprema za evaluaciju, a 80% za treniranje.

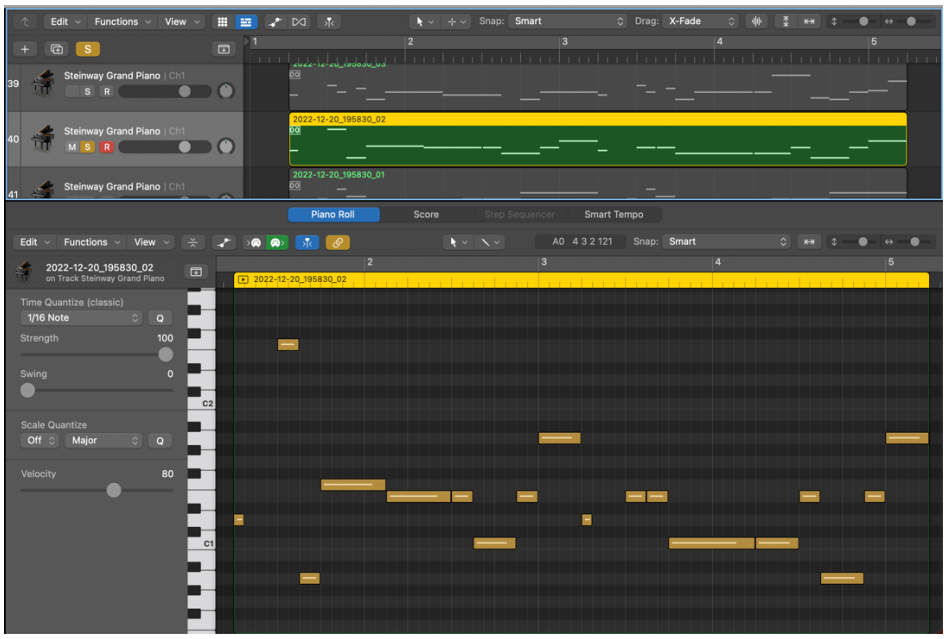
Korištenjem manjih serija i manjih slojeva u neuronskoj mreži znači da će model biti brže istreniran, ali i da će imati nešto lošije rezultate (19). Unatoč preporučenom smanjenju serije treniranje modela trajalo je nešto više od četrnaest sati i dobiven je model kojeg možemo spremiti u datoteku nalik paketu koja sadrži sve što je potrebno za pokretanje modela (engl. *bundle file*).

4.3. Rezultati i evaluacija

Za usporedbu različitih modela strojnog učenja proveden je upitnik o kojem ću nešto detaljnije napisati u kasnijem poglavlju. Za sada je potrebno samo saznati koliko je melodija koju generira model dobra za ovaj test. Procjena koliko je melodija kvalitetna svakako je subjektivna stvar, ali kvaliteta melodije donekle se može i objektivno vrednovati po njenoj kompleksnosti, po tome koliko dobro se može iskoristiti u pravom scenariju (u pjesmi) i koliko je lako pamtljiva (engl. *catchy*).

Jedna od pjesama koja se našla u skupu za treniranje je „Poison“ Alice Coopera. Pod uvjetom da u skup za treniranje ubacimo melodiju u njenoj cijelosti (od početka do kraja pjesme) ta glavna tema ponovi se vjerojatno preko 20 puta u trajanju pjesme. S druge strane imamo neke pjesme gdje se tema pojavi 3 ili 4 puta kroz pjesmu, što znači da „Poison“ ima unaprijed veliku prednost u odnosu na ostale pjesme.

Upravo to se i očitovalo u rezultatima - nakon generiranja deset melodija, svaka od njih je u dijelu svoje melodije imala cijelu melodiju pjesme „Poison“, što bi značilo da je imala jako velik utjecaj na cjelokupni trening mreže. Prilikom istraživanja drugih metoda generiranja melodija putem računala iz znatiželje sam poslušao nekoliko primjera glazbe koja je generirana prethodno spomenutom metodom varijacijskih autoenkodera (28). U jednoj od generiranih melodija jasno se može čuti pjesma „Super Freak“ Ricka Jamesa (poznatija po obradi „Can't Touch This“) (29) što uopće ne čudi obzirom na to koliko se puta ta tema ponovi u toj pjesmi.



Slika 3 - grafički prikaz meni osobno najbolje melodije generirane Magentom, melodija broj 11

Melodije koje ulaze u skup za treniranje svakako bi trebale biti trajanja duljeg od četiri takta - s tim trajanjem neke melodije tek završe jedan svoj ciklus, a neke se ponove dva ili čak četiri puta, što potencijalno znači da imaju četiri puta veći utjecaj na treniranje modela zbog svoje repetitivne prirode. U idućoj iteraciji iz istog skupa pjesama uzeo sam osam taktova glavne teme iz svake od pjesama i ponovio treniranje modela. Rezultati su bili svakako puno bolji, ali zbog prirode tako malog skupa podataka i zbog 20000 koraka u fazi treniranja, model je bio pretreniran. Zatim sam smanjio broj koraka na 500, a povećao veličine slojeva neuronske mreže s 64 na 128 što je dodatno poboljšalo rezultate te su melodije postale raznovrsnije. Bez obzira na veličinu skupa za treniranje, praktički je nemoguće točno procijeniti kakvoću trenirane mreže jer od dvadeset generiranih melodija, devetnaest ih može biti rubno neiskoristivih, a jedna može biti nevjerojatno dobra. S druge strane, u idućih dvadeset generiranih melodija može se desiti da ih je petnaest izvrsnih, iznadprosječno dobrih. Poanta

je da pravila nema - i s najboljim i najvećim skupom podataka možemo dobiti loše melodije, ali u tom slučaju barem imamo znatno manje šanse da se takvo nešto i dogodi.

5. Generiranje melodija pomoću genetskog algoritma

Istraživanjem raznih algoritama, inicijalna ideja bila je kreirati vlastiti model neuronske mreže s povratnim vezama tako da se može napraviti jasna i poštena usporedba s Melody RNN modelom. Na koncu, koju god metodu i koji god algoritam koristio, svaki od modela radit će jednu stvar - generirati melodije. Iako je genetski algoritam najčešće korišten za probleme optimizacije rješenja nekog problema, upravo je ta optimizacija ono što osobno tražim u problemu generiranja melodije.

5.1. Način primjene genetskog algoritma u generiranju melodija

Generiranje melodija korištenjem genetskog algoritma radit ću u Pythonu korištenjem genetskog algoritma iz (30). Kako implementacija genetskog algoritma obično započinje populacijom slučajnih jedinki, tako će genetski algoritam kreirati onoliko nasumičnih melodija koliko algoritmu zadamo u parametrima (zadana vrijednost je deset melodija). Pošto smo već ustanovili da baš i ne postoji neki objektivan način za ocjenjivanjem melodija, ovaj algoritam oslanja se na korisnika na način da mu dodjeljuje ulogu suca. Korisnik nakon svake melodije ocjenjuje tu melodiju ocjenom od 1 do 5. Algoritam će zatim zadržati dvije najbolje ocijenjene melodije, a preostale melodije će iskoristiti na način da će primijeniti križanje s najboljima i promijeniti neku od nota unutar melodije (mutacija). Na taj će način melodije s višom ocjenom opstati u nadi da će svaka sljedeća generacija biti sve bolja i bolja što se tiče kvalitete generiranih melodija. Duljina jedinke je u modelu kodirana pomoću formule: broj taktova * broj nota u taktu * broj bitova po noti (u ovom slučaju četiri). Nad tim binarnim zapisom se vrše operacije selekcije, križanja i mutacije, a nakon toga se pomoću funkcije `int_from_bits` svaka nota preslikava u svoj MIDI zapis (od 0 do 127).

5.1.1. Tehnički preduvjeti i dodatna glazbena teorija

Što se tehničkih zahtjeva tiče, također je korišten Python verzije 3.6.0. Od biblioteka korištena je pyo 1.0.1 biblioteka koja služi upravo da pretvori note nacrtane na računalu u zvuk kojeg zvučnici mogu reproducirati i melodiju koju možemo čuti. Zatim je korištena biblioteka MIDIUtil verzija 1.2.1 koja služi za upravljanjem MIDI datotekama, odnosno njihovim izvozom na lokalno računalo nakon generiranja u Pythonu. Biblioteka naziva Click (engl. ***C**ommand **L**ine **I**nterface **C**reation **K**it*) nije važna za sam proces generiranja melodija, ali puno znači u korisničkom sučelju naredbenog retka.

Nakon pokretanja programa moramo unijeti nekoliko parametara koji pobliže opisuju način generiranja inicijalne populacije i svojstva - to znači da će svaka melodija u inicijalnoj populaciji zadovoljavati sve postavljene parametre. Svaki od parametara ima svoju zadanu vrijednost tako da praktički nije potrebno unositi nikakve vrijednosti, ali su onda šanse da generiramo melodiju po vlastitom ukusu jako male.

Parametri su:

```
@click.option("--num-bars", default=8, prompt='Number of bars:', type=int)
```

Prvi parametar, *num-bars* je broj taktova kroz koje će se melodija generirati, tj. koliko dugo će melodija trajati (broj taktova direktno utječe na vrijeme izvođenja melodije).

```
@click.option("--num-notes", default=4, prompt='Notes per bar:', type=int)
```

Parametar *num-notes* je maksimalni broj nota po taktu, odnosno koliko će se najviše nota odsvirati u jednom taktu.

```
@click.option("--pauses", default=True, prompt='Introduce Pauses?', type=bool)
```

Parametar *pauses* je odabir želimo li pauze u melodiji - to je stvar osobnog ukusa i želje, postoje melodije poput već navedenih „Sweet Child O' Mine“ ili „Crazy Train“ koje nemaju pauze u melodijama, dok „Smooth Criminal“ Michaela Jacksona ili „Walk This Way“ Aerosmitha imaju pauze u svojim melodijama.

```
@click.option("--key", default="C", prompt='Key:', type=click.Choice(KEYS,
case_sensitive=False))
```

Četvrti parametar je ključ u kojem će melodija biti odsvirana (radi se o skupu nota iz kojeg se note biraju da bi međusobno imale smisla i zvučale dobro).

```
@click.option("--scale", default="major", prompt='Scale:',
type=click.Choice(SCALES, case_sensitive=False))
```

Zatim imamo peti parametar, a to je ljestvica. Ako se sjećate nešto glazbenog, znat ćete da postoje dur i mol akordi. To je izravno vezano uz ljestvicu u kojoj se melodija nalazi i intervale između nota ljestvice. Ljestvica može biti durska, molska, jonska, dorska, frigijska, lidijska, itd. Najčešće ćemo koristiti dursku ljestvicu za vesele melodije i molsku za tužne melodije - to je sve što moramo znati, a ako želimo da pjesma ima etno, istarski ili orijentalni prizvuk, krenut ćemo u istraživanje ljestvica i međuovisnost nota.

```
@click.option("--root", default=4, prompt='Scale Root:', type=int)
```

Parametar „Scale Root“ je visina tona željene ljestvice u željenom ključu (primjerice durska ljestvica u ključu C visine 4 označava C4 dursku ljestvicu frekvencije 261.63 Hz).

```
@click.option("--population-size", default=10, prompt='Population size:',
type=int)
```

Population size doslovno znači veličinu populacije, odnosno koliki broj melodija će se generirati u danoj seriji. Preporuka je odabrati paran broj zbog prirode algoritma križanja, na taj način će i u svakoj idućoj genetskoj seriji ostati jednak broj populacije. U suprotnom će se broj populacije smanjiti na prvi manji paran broj u idućoj seriji.

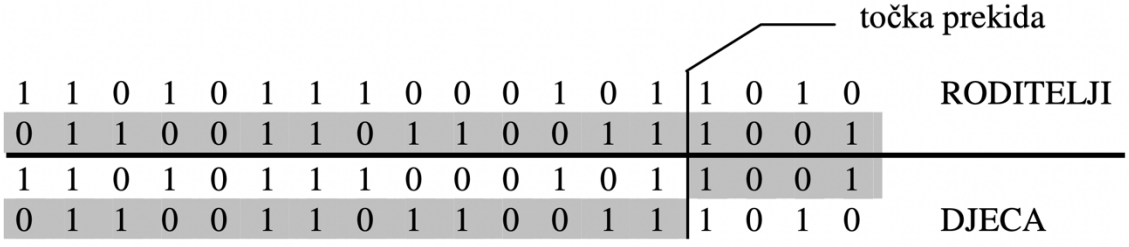
```
@click.option("--num-mutations", default=2, prompt='Number of mutations:',
type=int)
```

Broj mutacija utječe na broj nota koje će se promijeniti iz generacije u generaciju, mutacije se mogu dogoditi na bilo kojoj od pjesama unutar populacije.

```
@click.option("--mutation-probability", default=0.5, prompt='Mutations
probability:', type=float)
```

Parametar „vjerojatnost mutacije“ objašnjava sam sebe, kolika je vjerojatnost da će se mutacija dogoditi.

Korištena je funkcija križanja s jednom točkom prekida - to znači da se genetski kod u jednoj točki dijeli i križa s drugim roditeljem kako bi se krenulo u sljedeću generaciju. U procesu križanja sudjeluju dvije jedinke (roditelji), a u ovom slučaju nastaju dvije jedinke koje se nazivaju djeca. U ovom slučaju, uzima se početak jedne melodije iz populacije i kraj iz druge kako bi se dobila nova jedinka u novoj iteraciji.



Slika 4 - križanje s jednom točkom prekida (3)

Nakon što ispunimo sve parametre spremni smo za ocjenjivanje melodija koje nam genetski algoritam servira. Ovaj proces je zamoran i u principu dosadan jer slušamo melodije koje su više-manje slične onima koje prvotno označimo kao najboljima u populaciji. One označavaju temelj za daljnje populacije i za daljnju optimizaciju melodije. Dakle, da su te dvije melodije koje označimo kao najboljima u populaciji temeljno dobre melodije, tada bi cijeli

genetski algoritam imao smisla. Međutim, te melodije, kako ću i opisati u rezultatima, same po sebi ne zvuče dobro i kakve god izmjene, križanja i mutacije se dogode nad tim melodijama, to naprosto ne može zvučati dobro.

5.2. Rezultati

Jednom kada pokrenemo model, melodije koje on generira zvuče upravo kao i način na koji su generirane - nasumično. Jednom kada odaberemo najbolju melodiju od ponuđenih, mjesta za napredak ima jako malo. Koja god nota da se promijeni, gdje god da se napravi pauza, te melodije i dalje djeluju nasumično i relativno siromašno. Bez obzira na sve parametre koje smo zadali, šansa da melodija zvuči malo sofisticiranije jako je mala. Ako generiramo melodiju koja se proteže kroz dva takta, bilo bi odlično kada bi se ona mogla i reproducirati iznova, a u većini slučajeva generirane melodije ne zvuče dovoljno koherentno, smisleno i povezano kada se reproduciraju u petlji. Također, ako generiramo melodiju koja se proteže kroz četiri takta, u tom slučaju šanse za generiranjem nečega što ima smisla od početka do kraja stvarno su minimalne - više će zvučati kao skup nota koje odgovaraju parametrima i koje su u istom ključu i ljestvici, ali glazbeno nemaju neku preveliku poveznicu.

6. Upitnik

Za formalnu evaluaciju melodija kreiran je upitnik koji se sastoji od ukupno petnaest melodija koje su generirala tri različita modela. Prvi model je genetski algoritam, drugi model je Magentin Melody RNN, a treći model je također Melody RNN, ali koji je naučen na vlastitom skupu odabranih melodija opisanom u 4. poglavlju. Svaki model generirao je pet melodija, a zatim je svakoj melodiji dodijeljen nasumičan broj od 1 do 15 te su datoteke imenovane „Melodija 01“, ..., „Melodija 15“. Svaki od ispitanika imao je ulogu suca te se tražilo da ocijeni svaku od melodija ocjenom od 1 do 5. Tekstualni opisi tih ocjena glase ovako:

1 - melodija je neiskoristiva, kao da su note nasumično naslagane

2 - melodija je loša, ali naziru se dobri dijelovi iz kojih se može izvući inspiracija

3 - melodija je u redu - uz neke izmjene ima potencijala postati puno bolja

4 - melodiju malo dijeli da postane izvrsna

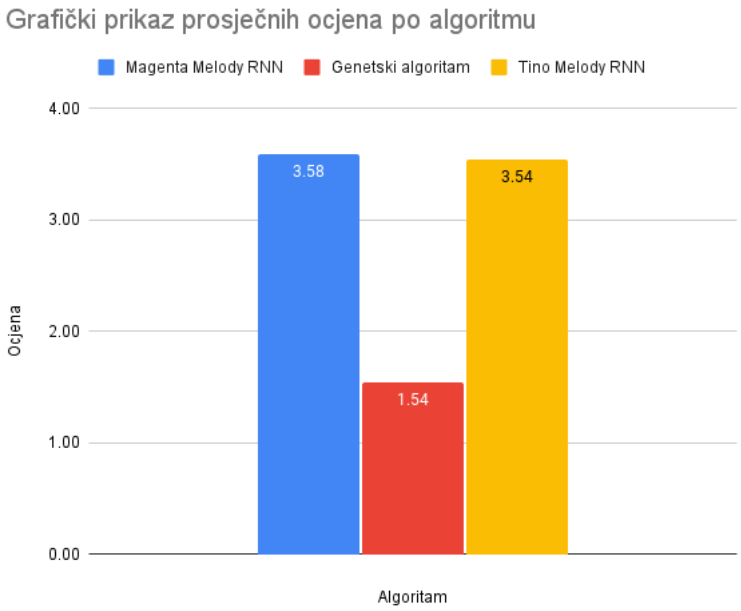
5 - melodija je izvrsna i kao takva mogla bi prerasti u pravu pjesmu.

Svi ispitanici imaju izvrsnu glazbenu podlogu, a neki od ispitanika su i izvođači u velikim hrvatskim glazbenim skupinama, kao što su Let 3, Neno Belan & Fiumensi, Putokazi, itd. Samim time što sam čuo melodije, mogu donijeti određena predviđanja za koja će se otkriti jesu li precizna i koliko. Rezultat koji očekujem je da će genetski algoritam biti puno lošije rangiran u odnosu na oba Magentina algoritma. Jedino što ostaje nepoznanica (a u tome malo igra i subjektivnost ispitanika vezano za žanr glazbe koji im se sviđa) je hoće li standardni Magenta *attention_rnn* model biti bolje, lošije ili jednako dobro ocijenjen kao model koji je naučen na vlastitom skupu melodija.

Upitniku je ukupno pristupilo 25 ispitanika, svaki ispitanik dobio je poveznicu na kojoj su se nalazile .mp3 datoteke s melodijama imenovane Melodija 01 do Melodija 15 i poveznicu na anketu u kojoj su unutar svakog pitanja imali Google Drive poveznicu za točno tu melodiju koju trenutno ocjenjuju. Nakon prikupljanja svih podataka, nazivima melodija dodao sam i nazive modela kojima su kreirani. Dva Melody RNN modela nazvao sam „Magenta Melody RNN“, a drugi koji sam ja trenirao nazvao sam „Tino Melody RNN“. Melodijama koje su generirane genetskim algoritmom dodao sam tekst „Genetski algoritam“. Zatim je u Google Sheets pomoću funkcije AVERAGE izračunata prosječna ocjena svake od melodija, pa tako melodija 1 ima prosječnu ocjenu 3.71, melodija 2 prosječnu ocjenu 3.96, itd. Najbolje ocijenjena melodija je melodija 14, generirana je modelom „Tino Melody RNN“ s prosječnom

ocjenom 4.04, dok je najlošije ocijenjena melodija 4, generirana genetskim algoritmom s prosječnom ocjenom 1.50.

Ipak, konačnu pobjedu odnosi model „Magenta Melody RNN“ koji je dosljedno i uzastopno za svaku generiranu melodiju dobivao visoke ocjene, ukupne prosječne ocjene 3.58, nakon njega slijedi model „Tino Melody RNN“ s ukupnom prosječnom ocjenom 3.54, a na kraju imamo genetski algoritam koji ostvaruje poražavajuću prosječnu ocjenu 1.54.



Slika 5 - grafički prikaz prosječnih ocjena svih melodija grupirano po algoritmu koji ih je generirao

Neki od ispitanika otišli su korak dalje i napisali kratki osvrt na cijeli eksperiment i izrazili svoje mišljenje o generiranim melodijama: „Moje je mišljenje da se melodije u kromatskoj ljestvici rijetko kada svide slušatelju bez pripadajuće harmonijske podloge - s podlogom one mogu biti ili savršenstvo ili potpuni promašaj. Sukladno tome, jednostavnije melodije više gode uhu i lakše ih je zamisliti u pjesmi - brže ćemo pronaći pripadajuću harmoniju za jednostavne nego za kompleksne melodije. Jednostavne melodije isto tako više podsjećaju na poznate pjesme i lakše je na njih nadodati bilo kakav tekst. To naravno ne znači da su jednostavne melodije nužno bolje, ali je veća šansa da bez konteksta i harmonijske podloge ostvare bolje rezultate nego one kompleksnije melodije.“

7. Zaključak

Strojno učenje koristi se u mnogim oblicima umjetnosti, od obrade fotografija, kreiranja slika, crteža do generiranja glazbenih djela. Navedeni su razni pristupi korištenja strojnog učenja, a zavisno o tome koji pristup se koristi za generiranje melodije imat će svoje prednosti i mane u krajnjem rezultatu.

U radu s genetskim algoritmom, modelu smo pružili sve glazbene parametre, od broja taktova, ključa, visine tona, ljestvice, dok Magenti nismo pružili niti jedan parametar. Svakako, Magenti smo pružili pozamašnu količinu melodija koje nam se sviđaju, ali pri generiranju i dalje nije pružen niti jedan parametar osim trajanja same melodije. Stoga se pitamo, koji model daje bolje rezultate? Jesu li bolji eksplicitno zadani parametri u genetskom algoritmu ili Magenta koja radi potpuno bez parametara?

Po mom mišljenju, nemoguće je uspoređivati modele na osnovu samih melodija jer se i najlošijem algoritmu može dogoditi da slučajno generira novi glazbeni hit 2023. godine, ali šanse da se to dogodi su zanemarive do te mjere da možemo sa sigurnošću reći da se to ipak neće dogoditi. Stoga možemo uspoređivati modele po tome koji ima veće šanse generirati kvalitetne melodije svaki put, bez obzira na parametre, iznova, na zahtjev korisnika. Tu je odgovor relativno jednostavan, a to je Magenta. Magenta generira melodije koje su sofisticiranije, imaju smisleniju razradu i završetak, uglavnom se mogu reproducirati u petlji i općenito su kompleksnije. Melodije generirane genetskim algoritmom dosta su ograničavajuće u smislu toga da se algoritam dosta čvrsto drži teme melodija koje smo označili najboljima u prvom prolazu melodija i radi minimalne preinake bez obzira na broj mutacija i šansu da se mutacija dogodi.

Kao zaključak, algoritmi strojnog učenja pokazali su stvarno veliki potencijal u generiranju glazbe i u poticanju kreativnog procesa u glazbenicima koji zatim mogu iskoristiti dobivene melodije za stvaranje nečeg jedinstvenog i s vlastitim potpisom. Iako s puno ograničenja, ta tehnologija već je započela toliko napredovati u području glazbe da se može reći da revolucionira glazbenu industriju i otvara nova vrata za glazbenike koji traže nove načine za glazbeno izražavanje.

8. Literatura

1. **Warner Bros. Entertainment.** YouTube. *YouTube.* [Mrežno] Warner Bros. Entertainment, 1930. https://www.youtube.com/watch?v=0jTHNBKjMBU.

2. **Henz, Martin, Lauer, Stefan i Zimmermann, Detlev.** COMPOzE — Intention-based Music Composition through Constraint Programming. *COMPOzE — Intention-based Music Composition through Constraint Programming.* Saarbrucken : University of Saarland, 1996.

3. **Golub, Marin.** Genetski algoritam: Prvi dio. *Genetski algoritam: Prvi dio.* 2004.

4. **Katoch, Sourabh, Singh Chauhan, Sumit i Kumar, Vijay.** A review on genetic algorithm: past, present, and future. *A review on genetic algorithm: past, present, and future.* Hamirpur : Springer Nature, 2020.

5. **MathWorks.** MathWorks. *MathWorks.* [Mrežno] https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html.

6. **Papadopoulos, George i Wiggins, Geraint.** A Genetic Algorithm for the Generation of Jazz Melodies. *A Genetic Algorithm for the Generation of Jazz Melodies.* Edinburgh : University of Edinburgh, 2000.

7. **Codes, Kie.** YouTube. *YouTube.* [Mrežno] 3. Travanj 2021. https://www.youtube.com/watch?v=nypJ3b4rMhE.

8. **Géron, Aurélien.** *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow.* Sebastopol : O'Reilly Media, Inc., 2019.

9. **Jiang, Tammy, Gradus, Jaimie L. i Rosellini, Anthony J.** Supervised machine learning: A brief primer. *Supervised machine learning: A brief primer.* Boston : an., 2020.

10. **Abolafia, Dan.** Magenta. *TensorFlow.* [Mrežno] Google, 10. Lipanj 2016. https://magenta.tensorflow.org/2016/06/10/recurrent-neural-network-generation-tutorial.

11. **Goodfellow, Ian J., i dr.** Generative Adversarial Nets. *Generative Adversarial Nets.* Montreal : Universite de Montreal, 2014.

12. **Colah, Chris.** Colah's Blog. *GitHub.* [Mrežno] 27. 8 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

13. **Brownlee, Jason.** A Gentle Introduction to Generative Adversarial Networks (GANs). *Machine Learning Mastery.* [Mrežno] 17. 6 2019. https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/.

14. **Li, Shuyu, Jang, Sejun i Sung, Yunsick.** Automatic Melody Composition Using Enhanced GAN. *Automatic Melody Composition Using Enhanced GAN.* Seoul : Department of Multimedia Engineering, Dongguk University, 2019.

15. **Simha, Anirudha.** Understanding TF-IDF for Machine Learning. *CapitalOne.* [Mrežno] CapitalOne, 6. 10 2021. https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/.

16. **Chollet, François.** *Deep Learning with Python.* Shelter Island : Manning Publications Co., 2018.

17. **Brownlee, Jason.** *Long Short-Term Memory Networks With Python.* Vermont, Victoria : Jason Brownlee, 2017.

18. **Sak, Hasim, Senior, Andrew i Beaufays, Francoise.** Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. *INTERSPEECH.* s.l. : Google, 2014.

19. **Google.** Magenta. *GitHub.* [Mrežno] Google. https://github.com/magenta/magenta/blob/main/magenta/models/melody_rnn/README.md.

20. **Briot, Jean-Pierre, Hadjeres, Gaetan i Pachet, Francois-David.** Deep Learning Techniques for Music Generation – A Survey. *Deep Learning Techniques for Music Generation – A Survey.* Paris : an., 2019.

21. **Ovčariček, Marin.** Prepoznavanje znamenki korištenjem neuronskih mreža. *Prepoznavanje znamenki korištenjem neuronskih mreža.* Zagreb : Sveučilište u Zagrebu, 2019.

22. **Waite, Elliot.** Magenta. *TensorFlow.* [Mrežno] Google, 15. Srpanj 2016. https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn.

23. **Raffel, Colin.** Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching.* San Francisco : Columbia University, 2016.

24. —. The Lakh MIDI Dataset. *The Lakh MIDI Dataset.* [Mrežno] https://colinraffel.com/projects/lmd/.

25. **Wikipedia.** MIDI. *Wikipedia.* [Mrežno] https://en.wikipedia.org/wiki/MIDI.

26. **MidiStudio.com.** CS. *CMU.* [Mrežno] MidiStudio.com, 22. Prosinac 1997. https://www.cs.cmu.edu/~music/cmsip/readings/GMSpecs_Patches.htm.

27. **Google.** Developers. *Google.* [Mrežno] Google. https://developers.google.com/protocol-buffers/.

28. **Tham, Isaac.** Generating Music Using Deep Learning. *Towards Data Science.* [Mrežno] 25. Kolovoz 2021. https://towardsdatascience.com/generating-music-using-deep-learning-cb5843a9d55e.

29. —. SoundCloud. *SoundCloud.* [Mrežno] Kolovoz 2021. https://soundcloud.com/isaac-tham-432434898/vae-good-1.

30. **Codes, Kie.** YouTube. *YouTube.* [Mrežno] 17. Kolovoz 2020. https://github.com/kiecodes/generate-music.