Automatic Music Composition Using Genetic Algorithm and Artificial Neural Networks

Projekat iz Računarske inteligencije
Matematički fakultet
Univerzitet u Beogradu

Katarina Milošević
mi19053@alas.matf.bg.ac.rs

Sep 2023

Sadržaj

1 Uvod

Muzika predstavlja široko i duboko polje samo po sebi, a komponovanje muzike je istinska umetnost i težak zadatak za pojedinca. Kada kompozitor stvara muzičke kompozicije, to čini sa dubokom namerom, razmišljanjem i svojom jedinstvenom kreativnošću. Dolazak automatizacije u proces komponovanja muzike je izazvao značajan interes i pažnju. Budući da je komponovanje muzike tradicionalno smatrano delom domena ljudske inteligencije, pojavljuje se fascinantna ideja primene veštačke inteligencije kako bi se postigao isti nivo izražajnosti i dubine u muzičkom stvaralaštvu.

S tim u vezi, motivacija za ovaj rad leži u želji da se istraže različite primene veštačke inteligencije u muzičkom stvaralaštvu. Cilj rada je generisanje inovativnih "muzičkih ideja"koje će se sastojati od sekvenci muzičkih nota i poslužiti kao osnova za dalje komponovanje muzičkih dela. Podaci koji će se koristiti obuhvataju raznovrsne melodije iz različitih muzičkih žanrova, a format u kojima su dati podaci je Ringtone Text and Transfer Language (RTTTL).

U okviru ovog rada predstavljeno je stvaranje muzike koristeći dva različita genetska algoritma. Prvi primenjuje Elman rekurentnu neuronsku mrežu, dok se drugi oslanja na pravila za stvaranje muzičkih kompozicija. Oba pristupa korišćena su kao sredstvo ostvarivanja cilja projekta - generisanje nove "muzičke ideje", tj. niza muzičkih nota koje služe kao osnova za komponovanje muzike.

2 Opis problema

Zadatak predstavlja proces transformacije ulaznih podataka u RTTTL formatu putem različitih metoda veštačke inteligencije, sa krajnjim ciljem generisanja kompozicija u MIDI formatu.

Sistem za generisanje muzike razvijen je korišćenjem evolutivnih algoritama i rekurentnih neuronskih mreža. Proces generisanja muzike je potpuno automatski i ne zahteva ljudsku interakciju tokom faze evolucije.

Dizajn projekta, uključujući vrstu podataka, arhitekturu neuronske mreže, strukturu hromozoma, genetske operatore i muzička pravila korišćena u genetskom algoritmu, razmatrani su u narednim sekcijama.

3 Metodologija

3.1 Skup podataka

Ulazni podaci koji se obrađuju dati su u formatu Ring Tone Text and Transfer Language (RTTTL). Kompanija "Nokia"razvila je RTTTL sa svrhom prenosa zvučnih signala na mobilnim telefonima. RTTTL format se sastoji od tri sekcije: naziva, podrazumevane vrednosti i podataka. Sekciju naziv predstavlja niz karaktera koji opisuje naziv melodije. Naziv može biti najveće dužine 10 karaktera i ne sme sadržati dvotačku ":". Sekcija podrazumevane vrednosti je skup vrednosti razdvojenih zapetama, gde svaka vrednost sadrži ključ i odgovarajuću vrednost za ključ, razdvojene znakom jednakosti -". Ključevi opisuju određene podrazumevane vrednosti koje treba koristiti tokom izvođenja melodije. Mogući ključevi su:

- `d` - trajanje
- `o` - oktava
- `b` - ritam, tempo

Sekcija podataka sastoji se od niza karaktera razdvojenih zapetama, gde svaki niz sadrži trajanje, notu, oktavu i opciono tačku (koja povećava trajanje note za pola).

Kao primer, razmotrimo melodiju ispod. Naziv melodije je *Barbiegirl*, a zatim sledi sekcija podrazumevane vrednosti koja govori da je podrazumevano trajanje melodije 4 (četvrtina), podrazumevana oktava je 5, a tempo melodije je 125 otkucaja u minuti. Nakon sekcije podrazumevane vrednosti sledi sekcija podataka, koja predstavlja sekvencu muzičkih nota melodije. Podrazumevane vrednosti se koriste kako bi se popunile nedostajuće vrednosti u sekciji podataka. Na primer, razmotrimo prvu notu u melodiji, tj. 8g#. To znači da je trajanje note 8 (osmina), a nota je g#, ali nema vrednosti za oktavu, pa se oktava za ovu notu uzima iz sekcije podrazumevane vrednosti, tj. 5. Dakle, ova muzička nota je 8g#5. Slično tome, podrazumevano trajanje se takođe može koristiti ako nedostaje u muzičkoj noti u sekciji podataka.

Primer elemenata skupa podataka.

Barbiegirl:d=4,o=5,b=125:8g,8e,8g,8c6,a,p,8f, 8d,8f,8b,g,8f,8e,p,8e,8c,f,c,p,8f,8e,g,f

3.2 Pretprocesiranje

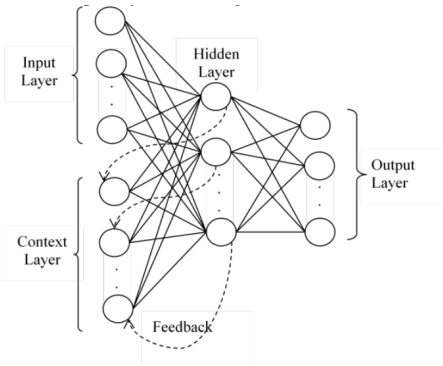
Da bi treniranje neuronske mreže na melodijama bilo uspešno, neophodno je da se melodija transformiše u format razumljiv za neuronsku mrežu. Svaki ulazni vektor za obuku mreže identifikuju trajanje, nota i oktava trenutne muzičke note. Dakle, u tu svrhu, ulaz treba prvo pretvoriti u sekvencu muzičkih nota ovog oblika.

Nakon što je svaka muzička nota u potrebnom formatu, sledeći korak u ovoj fazi je razdvajanje svake muzičke note na trajanje, notu i oktavu. Na primer, razmotrimo muzičku notu 16d#5. Nakon razdvajanja, *trajanje*=16, *nota*=d#, *oktava*=5. Potrebno je da svaka muzička nota u sekvenci melodije bude obrađena na ovaj način.

Izabrani skup podataka je Zip file of Mixed Tunes 2 (375 tunes), i na njemu je izvršen prethodno pomenut proces obrade podataka. Nakon obrade, od ukupno 375 melodija, preostalo je 227 ispravnih melodija, što je dalo 8275 ulaznih podataka za našu mrežu.

3.3 Neuronska mreza

Neuronska mreža korišćena u projektu je Elman rekurentna neuronska mreža, kako je prikazano na Slici 1. Ova mreža ima jedan sloj ulaznih podataka sa 23 neurona, skriveni sloj sa promenljivim brojem jedinica, sloj konteksta koji se sastoji od istog broja jedinica kao i skriveni sloj, a u koji se dovodi izlaz iz skrivenog sloja iz prethodnog vremenskog koraka, još jedan skriveni sloj u koji se dovodi izlaz iz prethodnog vremenskog koraka sa drugačijom aktivacionom funkcijom, i izlazni sloj sa 23 neurona.



Slika 1: Elman RNN

Ulazi u neuronsku mrežu su trajanje (duration), nota (note) i oktava (octave) za trenutni vremenski korak, dok su ciljni izlazi trajanje, nota i oktava za sledeću muzičku notu u sekvenci melodije. Dakle, u svakom koraku, neuronska mreža pokušava da predvidi sledeću muzičku notu u sekvenci.

Ulazni sloj je linearan i zato je odabran binarni ulazni vektor veličine 23x1.

Mogući broj trajanja je 6, broj nota je 13, uključujući notu za pauzu, i postoje 3 oktave koje su moguće, što ukupno iznosi 23. Kako bi se uzeli u obzir tačkasti zapisi (dotted notes), koristi se dodatni bit.

Prvih 6 bitova predstavljaju trajanje, sledećih 13 bitova notu, narednih 3 bita oktavu, a poslednji bit predstavlja tačkastu notu. Vrednost bita "1žnači da je određena nota aktivna u trenutnom vremenskom koraku, inače je taj bit postavljen na vrednost "0".

Primer kodiranog elemenata dataseta.

(8, 'b', 6, 1) Bismo predstavili kao:
[0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0.]

3.4 Genetski algoritam

3.4.1 Struktura hromozoma

U ovom projektu, svaka jedinka je predstavljena hromozomom, pri čemu je svaki hromozom povezan sa određenom vrednošću prilagođenosti (fitness). Struktura hromozoma prikazana je na slici 2. Svaki hromozom predstavlja pojedinačnu melodiju i sastoji se od više gena. Svaki gen, s druge strane, predstavlja muzičku notu i dodatno je podeljen na tri komponente: trajanje, nota i oktava.

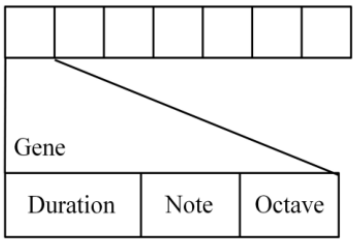
Da bismo to ilustrovali primerom melodije sa Slike 1, razmotrimo prvu muzičku notu u melodiji, koja glasi 8g#5.
Ovo se prevodi u prvi gen hromozoma na sledeći način:

- `Trajanje` - 8
- `Nota` - g#
- `Oktava` - 5

Slično tome, sledeća nota u melodiji, "8e5,"prevodi se u drugi gen hromozoma na sledeći način:

- `Trajanje` - 8
- `Nota` - e
- `Oktava` - 5

Ova struktura omogućava svakom hromozomu da predstavlja celu melodiju, a svaki gen unutar hromozoma kodira karakteristike muzičke note, uključujući njeno trajanje, ime note i oktavu.



Slika 2: **Struktura hromozoma**

3.4.2 Genetske operacije

U genetskom algoritmu (GA), na kraju svake generacije, genetski operatori, tačnije selekcija, mutacija i ukrštanje, deluju na skupu jedinki koji čine populaciju. Kao selekcijski operator korišćen je "roulette wheel selection"(ruletska selekcija). U ovom pristupu, nivo prilagođenosti (fitness) hromozoma se koristi kako bi se svakoj jedinki dodelila verovatnoća izbora.
Ako je f_i prilagođenost jedinke i u populaciji, njena verovatnoća da bude izabrana je $\pi_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$, gde je N broj jedinki u populaciji.

Kao operatori mutacije korišćeni su swap, scramble, inversion i replace. Swap mutacija se koristi za zamenu dve nasumično odabrane muzičke note. Operator scramble meša gene između dve nasumično odabrane tačke. Inversion operator invertuje hromozom između odabranih tačaka. Replace se koristi za zamenu nasumično odabrane note novom, nasumično generisanom notom.

Operatori ukrštanja koji su korišćeni uključuju one-point, two-point i uniform crossover. One-point ukrštanje bira nasumičnu tačku, seče dva različita genoma oko te tačke i stvara potomke koristeći te sečene delove. Two-point ukrštanje radi istu stvar, ali bira dve nasumične tačke. Uniform ukrštanje ne ograničava broj tačaka sečenja i ravnomerno raspoređuje sve gene roditelja nasumično među potomcima.

Takođe, verovatnoća za ukrštanje je postavljena na 0,7, a verovatnoća za mutaciju na 0,1. Ovi operatori su zatim primenjivani u skladu sa navedenim verovatnoćama.

3.4.3 Muzička pravila koja se primenjuju u genetskom algoritmu

1. **Pravilo Skale:** Ovo pravilo proverava da li dati melodi odgovara određena muzička skala, kao što je C-dur (C D E F G A B). Izabrana je bas C-dur skala.

2. **Pravilo Susednih Tonova:** Ono procenjuje da li je razlika između većine susednih tonova u melodiji veća od jednog polutona i da li su veliki skokovi ograničeni na manje od 4 polutona. Uzeta je vrednost maksimalnog broja koraka 2 i maksimalnog broja preskoka 4.

3. **Odnos između Pauza i Tona:** Pronalazi odnos između tonova i pauza u datoj melodiji. Odnos između tonova i pauza je podešen kao parametar. Izabrana vrednost je 1.5 sto odgovara negde oko 60% - 40%.

4. **Pravilo Ponavljanja Tonova:** Ako datu melodiju karakterišu ponavljanja tonova ili pauza, važno je da ne pređe određeni prag, kako melodija ne bi zvučala monotono i dosadno. Maksimalan broj ponavljanja tonova i/ili pauza može se postaviti kao parametar. Odlucila sam se za vrednost 2.

5. **Pravilo Globalne Distribucije Visine Tona:** Ovo pravilo proverava da li najniži i najviši ton u datoj melodiji padaju unutar određenih granica. Granica je izražena kao broj polutona. Granica koja je koriscenja u algoritmu je $+-12$.

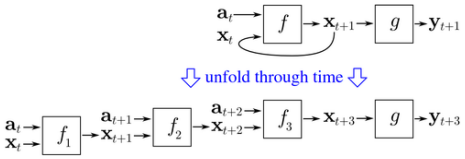
4 Algoritmi

4.1 Alogoritam treniranja Elman RNN

Propagacija unazad kroz vreme (BPTT) je tehnika zasnovana na gradijentima koja se koristi za obuku rekurentnih neuronskih mreža (RNN). To je varijacija algoritma propagacije unazad prilagođena za sekvencijalne podatke i posebno je pogodna za RNN-ove zbog njihove sposobnosti obrade sekvencijalnih informacija.

U BPTT-u se težine i pristrasnosti mreže ažuriraju računanjem gradijenata gubitka funkcije u odnosu na parametre mreže tokom sekvence vremenskih koraka. Ti gradijenti se zatim koriste za prilagođavanje parametara putem gradijentnog spusta ili sličnih metoda optimizacije.

BPTT razvija RNN kroz vreme kako bi stvorio računarski graf koji predstavlja zavisnosti između vremenskih koraka. Ovo omogućava algoritmu da izračuna gradijente za svaki vremenski korak i ažurira parametre mreže prema tome. Proces propagacije unazad se izvodi kroz razvijenu mrežu, a gradijenti se akumuliraju i koriste za ažuriranje težina.



Slika 3: **Unfold through time**

Za potrebe algoritma korišćena je custom loss funckija. Ideja za njenu uportebu je da će loss biti manji u slučaju da naš vektor dužine 23 podelimo na manje vektore koji ustvari predstavlaju kodirane dužine, note, oktave i taču i za svaki deo posebno izračunamo loss u zavisnosti od klasifikacionog problema koji predstavlaju. (recimo za tačku koja ima vrednost 0/1, primenićemo binarnu unakrsnu entropiju, dok za note, dužine i oktave pimenjujemo kategoričku unukrsnu entropiju).

Pošto postlednji layer ima za aktivacionu funkciju linearnu, moramo koristi opciju funkcija with_logits.

```
def split_and_calculate_loss(target, out):
# Idea is to split tensor in format that was first introduced,
and then apply loss on parts of tensor
# So i split it in 6, 13, 3, 1

# Split the tensors
target_splits = tf.split(target, [6, 13, 3, 1], axis=-1)
out_splits = tf.split(out, [6, 13, 3, 1], axis=-1)

# Calculate loss for each split
loss_splits = []
for i, (t, o) in enumerate(zip(target_splits, out_splits)):
    # labels = tf.cast(tf.squeeze(t, axis=-1), dtype=tf.float32)
    # logits = tf.cast(tf.squeeze(o, axis=-1), dtype=tf.float32)
    if i == 3:  # The last split with size 1
        loss = tf.nn.sigmoid_cross_entropy_with_logits(
        labels=tf.cast(tf.squeeze(t, axis=-1), dtype=tf.float32),
        logits=tf.cast(tf.squeeze(o, axis=-1), dtype=tf.float32))
    else:
        loss = tf.nn.softmax_cross_entropy_with_logits(labels=t, logits=o)
    loss_splits.append(loss)

# Combine the losses, use mean
combined_loss = tf.reduce_mean(tf.stack(loss_splits, axis=-1))

return combined_loss
```

4.2 Genetski alogoritam koji koristi istraniranu Elman RNN

1. Početna populacija melodija se stvara u multiplikacijama broja 2 iz praktičnih razloga i formira se korišćenjem slučajnog niza muzičkih elemenata.

2. Generišu se hromozomi kako bi se formirala trenutna populacija trenutne generacije.

3. Evaluira se fitnes različitih hromozoma korišćenjem obučene RNN. Svaki par vrednosti se koristi kao ulazna vrednost za RNN, a naredni par se koristi za poređenje sa izlaznom vrednošću, pri čemu se računa razlika. Sve razlike za sve parove se akumuliraju kako bi se dobila krajnja vrednost greške, koja se koristi za izračunavanje fitnesa jedinke.

$$f_j = \left(\sum_{i=1}^{m} t_i - a_i \right) \quad j = 1, 2, \ldots, n$$

Gde je m ukupan broj parova vrednosti, t_i je ciljana vrednost, tj. naredni uzastopni par, a_i je stvarna vrednost i n je broj jedinki.

4. Ako postoji hromozom koji je dovoljno dobar, tj fitness mu je manji od zadatog fitnesa, završava se ovde i taj hromozom se izdvaja kao melodija trenutne generacije. Inače, prelazi se na korak 5.

   Ovde se upoređuju svi f_j koji su ranije izračunati sa željenom minimalnom vrednošću kondicije $F_{\min}$.

   Ako $f_j < F_{\min}$, čuva se vrednost fitnesa i odgovarajuća jedinka.

   Izdvajaju se melodije za sve takve sačuvane kondicije i genetski algoritam se zaustavlja.

5. Vrši se selekcija pomoću selekcije na ruletu.

6. Vrše se mutacije i ukrštanja.

7. Idi na korak 2.

4.3 Genetski alogoritam koji koristi muzička pravila

1. Genetski algoritam za melodije koristi jednostavna muzička pravila iz teorije muzike. Fitnes funckiaj se sastoji od provere validnosti ovih pravila u vezi sa tim kako se početne melodije razlikuju od idealnih.

2. Kreirajte početnu populaciju melodija. Početna populacija se stvara korišćenjem slučajnog niza muzičkih elemenata.

3. Generišite hromozome kako biste formirali trenutnu populaciju trenutne generacije.

4. Evaluirajte fitness različitih hromozoma korišćenjem muzičkih pravila. Sva pravila računaju odstupanje između generisane melodije i navedenih pravila. Niža vrednost kondicije znači manje odstupanje, što znači bolju melodiju.

   Kondicija je data matematički kao:

$$f_j = \frac{1}{m} \sum_{i=1}^{m} errorCount_i; \quad j = 1, 2, \ldots, n$$

   Gde je m broj muzičkih pravila i n broj individualnih hromozoma.

5. Ako je bilo koji hromozom dovoljno kondicioniran, zaustavite se ovde i izdvojte taj hromozom kao melodiju trenutne generacije, inače pređite na korak 5.

   a. Uporedite svaki f_j koji je ranije izračunat sa željenom minimalnom vrednošću kondicije $F_{\min}$.

   b. Ako je $f_j < F_{\min}$, sačuvajte vrednost kondicije i odgovarajuću individuu.

   c. Izdvojte melodije za sve sačuvane kondicije i zaustavite Genetski Algoritam.

6. Izvršite selekciju korišćenjem selekcije na ruletu.

7. Izvršite mutaciju i ukrštanje.

8. Idite na korak 2.

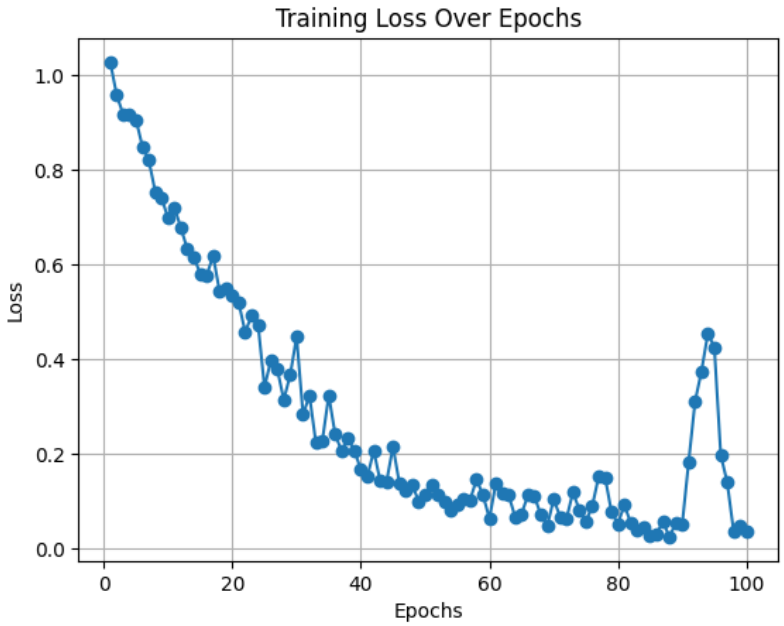
5 Rezultati

Rezultati dobijeni kao deo projekta koji razmatrani su, uključujući obuku neuronske mreže, genetski algoritam koji koristi neuronsku mrežu kao procenitelja fitnesa i genetski algoritam koji koristi muzička pravila kao fitness funkciju.

5.1 Rezultati treniranja Elman RNN

Treniranje neuronske mreže je izvršeno za 100 epoha, dobijen je loss od 0.02, kao optimizator korišćen je Adam sa learning rate-om 0.001. Ideja je bila napraviti sekvence podataka određene dužine u našem slučaju 50, ulazni podaci znači predstavljaju 50 uzastopnih nota a potrebno je predvideti 51 notu.

Ovako se menjala loss funckija kroz epohe.



Slika 4: **Loss funckija kroz epohe**

5.2 Rezultati genetskog algoritma u slučaju korišćenja RNN

Pošto je izvršavanje genetskog algoritma bilo iscrpno, i trajalo je oko 3.5h za sekvencu dužine 50, veličinu populacije 50 i broj generacija 5000. Najmanja vrednost fitnesa koja je pronađena je velika i iznosi 9. Iako ovako velika vrednost fitnesa, nije naznačila da naš genetski neće proizvesti neke od zanimljivih i prijatnih melodija.

5.3 Rezultati genetskog algoritma u slučaju korišćenja muzičkih pravila

Pokrenut je veći broj izvršavanja genetskog algortima sa mužičkim pravilima, tj za različitite dužine sekvenci, veličine populacije i brojeve sekvenci.

Najmanja vrednost fitnesa koja je ovde pronađena je

6 Zaključak

Treniranje ovakve neuronske mreže je vremenski iscrpan zadatak i potrebno je pronaći odgovarajuću vrednost hiperparametara da bi rezultat bio zadovoljavajući.

Generisane melodije koje su koristile RNN kao evaulator u fitness funkciji su imale nekoliko manjih grešaka. Vreme izvršavanja ovog pristupa bilo je u satima, ali opet generisane melodije su složenije od onih koje su generisane drugim genetskim algoritmom.

Genetski algoritam koji je koristio muzička pravila se u drugu ruku izvršavao svega nekoliko minuta. Generisane melodije su bile jednostavnije i prijatne. Korišćenjem većeg broja muzičih pravila uverena sam da bi se dobile jos složenije melodije.

7 Literatura

[1] Materijali iz Računarske inteligencije
[2] Various Artificial Intelligence Techniques For Automated Melody Generation
[3] Generate music with an RNN
[4] Backpropagation through time