UNIWERSYTET IM. ADAMA MICKIEWICZA W POZNANIU
WYDZIAŁ MATEMATYKI I INFORMATYKI

Cezary Adam Pukownik

Kierunek: Analiza i przetwarzanie danych
Numer albumu: 444337

Generowanie muzyki przy pomocy głębokiego uczenia

Music generation with deep learning

Praca licencjacka
napisana pod kierunkiem
dr hab. Tomasza Góreckiego

POZNAŃ 2020

Poznań, dnia .....

OŚWIADCZENIE

Ja, niżej podpisany Cezary Pukownik, student Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt: "Generowanie muzyki przy pomocy głębokiego uczenia", napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w wersji drukowanej jest całkowicie zgodny z egzemplarzem pracy dyplomowej w wersji elektronicznej.

Jednocześnie przyjmuję do wiadomości, że przypisanie sobie, w pracy dyplomowej, autorstwa istotnego fragmentu lub innych elementów cudzego utworu lub ustalenia naukowego stanowi podstawę stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego.

$[TAK]^\star$ - wyrażam zgodę na udostępnianie mojej pracy w czytelni Archiwum UAM

$[TAK]^\star$ - wyrażam zgodę na udostępnianie mojej pracy w zakresie koniecznym do ochrony mojego prawa do autorstwa lub praw osób trzecich

*Należy wpisać TAK w przypadku wyrażenia zgody na udostępnianie pracy w czytelni Archiwum UAM, NIE w przypadku braku zgody. Niewypełnienie pola oznacza brak zgody na udostępnianie pracy.

.....

Spis treści

Streszczenie

Abstract

Wstęp

Uczenie maszynowe w ostatnich latach mocno zyskało na popularności. Zastosowania i możliwości różnych algorytmów uczenia maszynowego czasami przekraczają nasze wyobrażenie o tym co komputer może zrobić. Niektóre aplikacje potrafią wręcz zaskoczyć użytkowników tym co potrafią zrobić. Wśród takich aplikacji znajdują się takie, które potrafią przewidywać następne wartości akcji giełdowych, rozpoznawać na filmie obiekty w czasie rzeczywistym, czy nawet prowadzić samochód. Algorytmy wyuczone proponują nam spersonalizowane reklamy, czy produkty na podstawie naszych upodobań. Najczęstsze zastosowania dotyczą przetwarzania obrazów lub tekstu, natomiast zastosowania w przetwarzaniu muzyki są niszowe i rzadziej spotykane.

Celem tej pracy jest stworzenie modelu sieci neurowej, którego zadaniem bedzie generowanie krótkich multiinstrumentalnych klipów muzycznych.

W pierwszej częsci swojej pracy przedstawię podstawowe koncepcje związane z muzyką oraz sposobami jej reprezentacji. Następnie opiszę w jaki sposób działają sieci neuronowe, jak się uczą oraz podstawowe architektury sieci, które pomogą zrozumieć model który wykorzystałem.

Następnie przedstawię koncepcję działania modelu, jakie idee stały za wyborami, które podjąłem w projektowaniu sieci. W szczegółowy sposób opiszę sposób ekstrakcji danych tak aby mogły być one wykorzystane przez model. Opiszę architekturę którą wybrałem oraz przedstawię i opisze fragmenty kodu w języku python.

W kolejnym rozdziale skupimy się na rezultatach pracy, przedstawie zalety i wady modelu. Przeprowadze analizę jakie muzyczne koncepcje model się nauczył na podstawie danych oraz doprowadzę do ostatecznej konkluzji czy wygenerowana muzyka może być przyjemna dla odbiorcy.

Wprowadzenie do sieci neuronowych

Aby lepiej zrozumieć w jaki sposób odpowiednio skonstruowane sieci neuronowe potrafią sprostać takiemu zadaniu jak generowanie muzyki, w tym rozdziale przedstawię od podstaw zasady działania sieci neuronowych. Opiszę w jaki sposób można od regresji liniowej przejść do prostych sieci oraz w jaki sposób uczy się sieci neuronowe. Ostatecznie przedstawię architektury, które wykorzystałem w projekcie.

1.1. Regresja liniowa

Podstawą wszystkich sieci neuronowych jest regresja liniowa. W statystyce wykorzystywana, aby wyjaśnić liniowe zależności między zmiennymi.

Prosty model regresji liniowej dla jednej zmiennej można opisać wzorem [1]

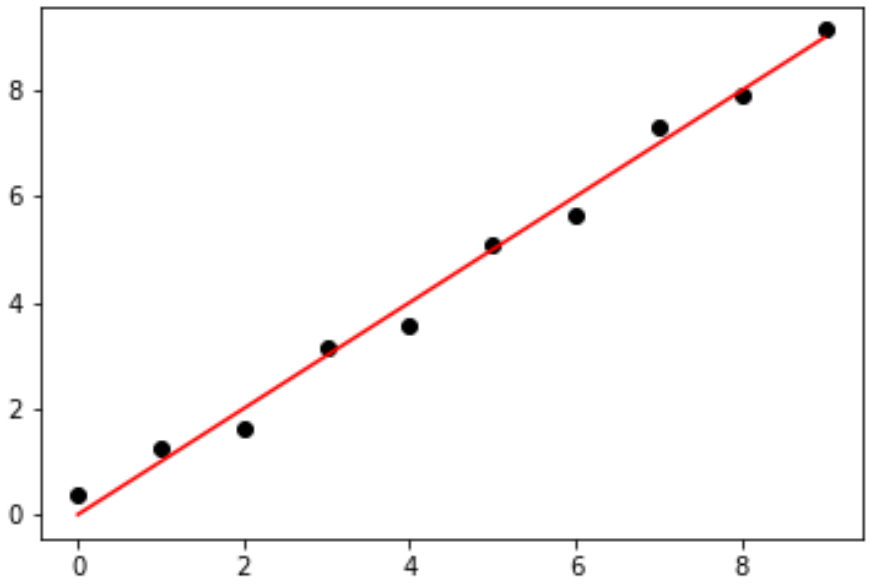
$$y = ax + b + \epsilon,$$

gdzie
— y jest zmienną objaśnianą,
— x jest to zmienna objaśniająca,
— a jest parametrem modelu,
— b jest wyrazem wolnym modelu,
— ϵ jest składnikiem losowym.

Zadaniem jest znalezienie takiego parametru $a \in \mathbb{R}$ oraz wyrazu wolnego $b \in \mathbb{R}$, aby dla znanych wartości $x \in \mathbb{R}$ oszacowanie zmiennej objasnianej $\hat{y} \in \mathbb{R}$ najlepiej opisywała zmienną objasnaną $y \in \mathbb{R}$. Tak zdefiniowany model opisuje zmienną y z dokładnością do składnika losowego. W praktyce oznacza to, że szacowane modele będą przybliżeniem opisywanych zależności.

Wartość zmiennej objaśnianej y można również opisać za pomocą wielu zmiennych objaśniających. Wtedy dla zmiennych objaśniającej $x_1, x_2, ..., x_n \in \mathbb{R}$ szukamy parametrów $\theta_1, \theta_2, ..., \theta_n \in \mathbb{R}$. Otrzymany w ten sposób model nazywany jest również hipotezą i oznaczamy go $h(x)$.

[1] Statystyka, Mieczysław Sobczyk s.179



Rysunek 1.1: Regresja liniowa jednej zmiennej

$$h(x) = b + \theta_1 x_2 + \theta_2 x_2 + ... + \theta_n x_n + \epsilon = b + \sum_{i=1}^{n} \theta_i x_i + \epsilon.$$

Rysunek 1.1 przedstawia przykładowy model regresji liniowej jednej zmiennej, dopasowany do zbioru.

1.2. Uczenie modelu

Celem uczenia modelu jest znalezienie ogólnych parametrów, aby model dla wartości wejściowych x zwracał wartości predykcji $\hat{y}$ najlepiej opisujące całe zjawisko według pewnego kryterium. Formalnie, aby suma wszystkich różnic między predykcją a rzeczywistością była najmniejsza.

$$\text{błąd} = \sum_{i=1}^{m} |\,\text{predykcja} - \text{rzeczywistość}\,|$$

, gdzie $m \in \mathbb{N}$ jest wielkością zbioru danych jakim dysponujemy. Minimalizując bład dla modelu jesteśmy wstanie znaleźć przybliżenie funkcji $h(x)$.

1.2.1. Funkcja kosztu

W tym celu używa się funkcji $J_\theta(h)$, która zwraca wartość błędu między wartościami $h(x)$ oraz y dla wszystkich obserwacji. Taka funkcja nazywana jest funkcją kosztu (*cost function*).

Dla przykładu regresji liniowej funkcją kosztu może być błąd średniokwadratowy (*mean squared error*). Wtedy funkcja kosztu przyjmuje postać:

$$J_\theta(h) = \frac{1}{m}\sum_{i=1}^{m}(y_i - h(x_i))^2$$

.

Przy zdefiniowanej funkcji kosztu proces uczenia sprowadza się do znalezienia takich parametrów funkcji $h(x)$, aby funkcja kosztu była najmniejsza. Jest to problem optymalizacyjny sprowadzający się do znalezienia globalnego minimum funkcji.

1.2.2. Znajdowanie minimum funkcji

Aby znaleźć minimum funkcji f możemy skorzystać z analizy matematycznej. Wiemy, że jeśli funkcja f jest różniczkowalna to funkcja może przyjmować minimum lokalne, gdy $f'(x_0) = 0$ dla pewnego x_0 z dzieniny funkcji f. Dodatkowo jeśli istanieje otoczenie punktu x_0, że dla wszystkich punktów z tego otoczenia spełniona jest nierówność:

$$f(x) > f(x_0)$$

to znaleziony punkt x_0 jest minimum lokalnym. W teorii należałoby zatem wybrać taką funkcję kosztu, aby była różniczkowalna. Obliczyć równanie $J'_\theta(h) = 0$, następnie dla otrzymanych wyników sprawdzić powyższą nierówność oraz wybrać najmniejszy wynik ze wszystkich[2]. W praktyce rozwiązanie takie równania ze względu na jego złożoność może się okazać niewykonalne. Aby rozwiązać ten problem powstały inne metody, które pozwalają szukać ekstremów funkcji, jednak nigdy nie będziemy mieli pewności, że otrzymany wynik jest minimum globalnym funkcji kosztu.

1.2.3. Metody gradientowe

Metody gradientowe (*gradient descent*) są to iteracyjne algorytmy służące do znajdowania minimum funkcji. Aby móc skorzystać z metod gradientowych analizowana funkcja musi być ciągła oraz różniczkowalna. Sposób działania ich można intuicyjnie opisać w nastepujących krokach.

1. Wybierz punkt początkowy.

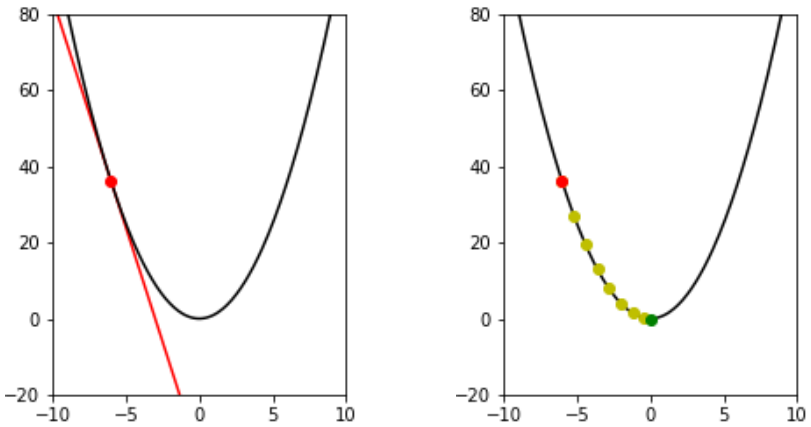
[2] źródlo: Analiza matematyczna, Krysicki Włodarski, s.187

2. Oblicz kierunek, w którym funkcja maleje.

3. Przejdź do kolejnego punktu zgodnie obliczonym kierunkiem o pewną małą odległość.

4. Powtarzamy, aż osiągniemy minimum funkcji.

Wizualizację algorytmu została przedstawiona na rysunku 1.2.



(a) Wyznaczenie gradientu (b) Iteracja kolejnych punktów

Rysunek 1.2: Wizualizacja algorytmu gradientu prostego

Dla funkcji $h(x)$ należy ustalić wartość początkową Θ_0 dla wszystkich parametrów θ_1 ... θ_n.

$$\Theta_0 = [\theta_1, \theta_2, ..., \theta_n]$$

Następnie policzyć wszystkie pochodne cześciowe $\frac{\partial J_\theta(h)}{\partial \theta_i}$. Otrzymamy w ten sposób gradient $\nabla J_\theta(h)$, gdzie

$$\nabla J_\theta(h) = \left[\frac{\partial J_\theta(h)}{\partial \theta_1}, \frac{\partial J_\theta(h)}{\partial \theta_2}, ..., \frac{\partial J_\theta(h)}{\partial \theta_n}\right]$$

Następnie obliczyć element Θ_{k+1} ze wzoru

$$\Theta_{k+1} = \Theta_k - \alpha \nabla J_\theta(h),$$

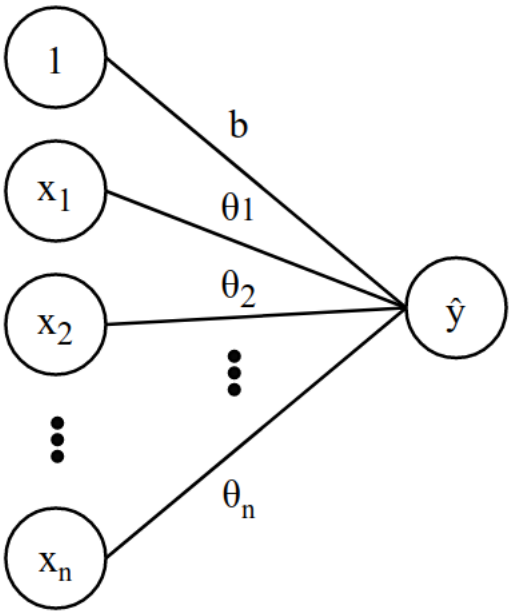
gdzie $\alpha \in \mathbb{R}$ jest współczynnikiem uczenia (*learning rate*). Proces ten należy powtarzać do pewnego momentu. Najczęściej z góry określoną liczbę razy lub do momentu, gdy uzysk funkcji kosztu spowodowany nastepną iteracją jest

mniejszy niż ustalona wartość. Otrzymany w ten sposób wektor parametrów Θ_k jest wynikiem algorytmu.[3]

Wykorzystując metody gradientowe otrzymujemy wyuczony model. Parametry θ_i modelu $h(x)$ zostały ustalone w taki sposób, aby błąd między predykcją a rzeczywistością był najmniejszy.

1.3. Regresja liniowa jako model sieci neuronowej

Omawiany model regresji możemy zapisać w sposób graficzny tak jak przedstawiono na rysunku 1.3.



Rysunek 1.3: Regresja liniowa jako model sieci neuronowej

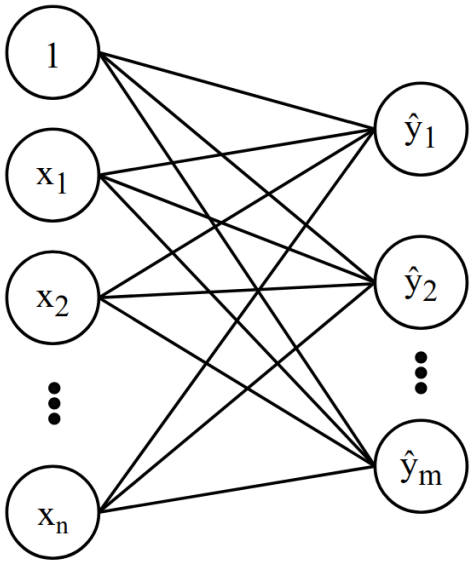
Każdy węzeł z lewej strony reprezentuje zmienną objaśniającą x_i. Połączenia nazywane są wagami i reprezentują one parametry θ_i. Węzeł z prawej strony oznaczony jako $\hat{y}$ jest sumą iloczynów wag oraz wartości węzłów z prawej strony. Wtedy

[3] Deep Learning techniques for music geneation - A survay s.44

$$\hat{y} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} b & \theta_1 & \theta_2 & \dots & \theta_n \end{bmatrix} = b + x_1\theta_1 + x_2\theta_2 + \cdots + x_n\theta_n = b + \sum_{i=1}^{n} x_i\theta_i$$

co jest równoważne omawianemu modelowi regresji liniowej. Węzły sieci nazywane są neuronami, a wyraz wolny modelu b nazywany jest biasem (*bias*).

W łatwy sposób możemy rozbudować ten model do regresji liniowej wielu zmiennych. Predykcją modelu nie będzie jak do tej pory jedna wartość $\hat{y}$, tylko wektor wartości $\hat{y_1}, \hat{y_2}, \dots, \hat{y_m}$, który oznaczać bedziemy jako $\hat{Y}$. Model ten zostal przedstawiony na rysunku 1.4.



Rysunek 1.4: Regresja liniowa wielu zmiennych jako model sieci nauronowe.j

Dla uogólnienia pojedyncze wagi modelu zapisywać będę jako w_{nm}, natomiast macierz wag jako W. Algebricznie zapisalibyśmy ten model jako

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{12} & \dots & x_{nm} \end{bmatrix} \begin{bmatrix} b_1 & w_{11} & w_{12} & \dots & w_{1n} \\ b_2 & w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_m & w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix} = \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_m(x) \end{bmatrix} = \begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y_m} \end{bmatrix}$$

$$b + XW = \hat{Y}$$

gdzie n jest liczbą zmiennych niezależnych, m jest liczbą zmiennych zależnych, X jest rozszerzonym do macierzy o rozmiarach m x n wektorem zmiennych objaśniających, w taki sposób że $x_{i1} = x_{i2} = \cdots = x_{in}$ dla $i = 1, 2, ..., m$, W jest macierzą wag o rozmiarach n x m, natomiast b jest sumą wyrazów wolnych $b_1, ..., b_m$. Możemy zauważyć, że model dla wielu zmiennych jest wieloma modelami dla jednej zmiennej, gdzie każdy model operuje na tych samych danych wejściowych. Taki model może być uznany za sieć neuronową i nazywany jest perceptronem.

1.4. Funkcje aktywacji

Omawiany model służy rozwiązywaniu problemu regresji, ponieważ wartości predykcji nie są uregulowane i mogą przyjmować wartości z $\mathbb{R}$. W celu przekształcenia tego modelu, aby móc go wykorzystać do rozwiązania problemu klasyfikacji, należy dodatkowo na otrzymanym wektorze $\hat{Y}$ wykonać pewną funkcję, która przekształci wynik. W tym celu używamy funkcji aktywacji (*activation function*). Istnieje wiele różnych funkcji aktywacji, a każda posiada inną charakterystykę i wpływ na model. Najpopularnijeszą grupą fukcji są funkcje sigmoidalne (*sigmoid functions*). Jedną z nich jest funkcja logistyczna (*logistic curve*) o wzrorze

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

oraz wykresie przedstawionym na rysunku 1.5

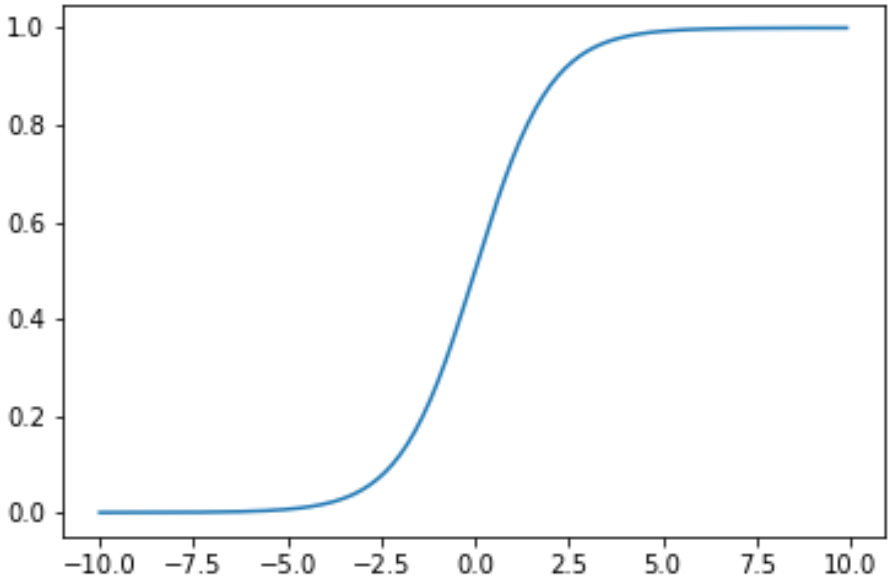
Funkcja logistyczna ma pewne użyteczne właściwości, które pozwolą kontrolować wartości węzłów oraz rzutować wartości z całego $\mathbb{R}$ do wartości z przedziału $(0, 1)$. Dzięki tej właściwości funkcja logistyczna jest często używana, aby otrzymać prawdopodobieństwo wystąpienia pewnego zdarzenia. Dodatkowo funkcja logistyczna szybko przyjmuje wartości skrajne, co oznacza że dla bardzo dużych wartości ujemnych i bardzo dużych wartości dodatnich funkcja staje się mało wrażliwa na zmiany wartości wraz ze zmianą wartości argumentu. [4]

W ten sposób możemy w łatwy sposób zmienić model regresji liniowej na model regresji logistycznej.

$$\sigma(b + XW) = \hat{Y}$$

W dalszych cześciach pracy, kiedy bedę używał funckji aktywacji nie wskazując na konkretną funckcję, będe wykorzystywał oznaczenie $AF(x)$.

[4] Deep Learning Book, s.66



Rysunek 1.5: Funkcja logistyczna

1.5. Głębokie sieci neuronowe

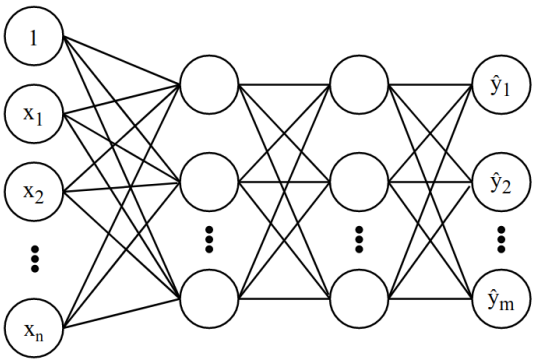
Model omawiany wcześniej może posłużyć jako podstawowy element do budowania bardziej skomplikowanych modeli. Aby to zrobić, należy potraktować otrzymany wektor $\hat{Y}$ jako wektor wejściowy do następngo podstawowego modelu. Składając ze sobą wiele perceptronów w jeden model, tworzymy warstwy (*layers*) sieci neuronowej.

Wyróżniamy trzy rodzaje warstw:

— warstwę wejściową (*input layer*), która jest pierwszą warstwą modelu,
— warstwę wyjściową (*output layer*), która jest ostatnią wartstwą modelu,
— wartwy ukryte (*hidden layer*), które są warstwami pomiędzy warstwą wejsciową oraz wyjściową.

Na rysunku 1.6 przedstawiono przykład posiadający warswę wejściową, dwie wartswy ukryte oraz wartswę wyjsciową.

Tego typu modele są głębokimi sieciami neuronowymi (*deep neural networks*). Istnieje wiele różnich architektur głębokich sieci neuronowych, które wykorzystują te podstawowe koncepcje i rozszerzają je o dodatkowe warstwy, połącznia, funkcje aktywacji czy neurony o specjalnych właściwościach.



Rysunek 1.6: Przykład modelu sieci neuronowej

1.5.1. Jednokierunkowe sieci neuronowe

Jednokierunkowe sieci neuronowe (*feedforward neural networks*) są to najprostrze sieci neuronowe, które wprost czerpią z omówionych wcześniej podstawowych wartsw. Możemy się również spotkać z nazwą wielowarstwowy perceptron (*multi layer perceptron - MLP*) ze względu na fakt, że jest zbudowany z wielu perceptronów zaprezentoanych w części 1.3. Działają one w taki sposób, że zasila się je danymi do warstwy wejściowej, następnie sukcesywnie wykonuje się obliczenia do momentu dotarcia do końca sieci. Każdy krok z warstwy $k-1$ do warstwy k obliczany jest zgodnie ze wzorem [5]

$$X_k = AF(b_k + W_k X_{k-1})$$

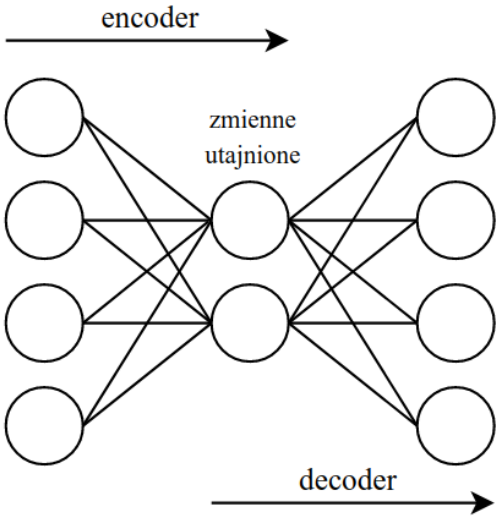
Propagacja wsteczna błędu

Kiedy używamy jednokierunkowych sieci neuronowych, zasilamy je danymi wejściowymi x ostatecznie otrzymując predykcję $\hat{y}$. Taki sposób działania nazywa się propagcją wprzód (*foreward propagation*). Podczas uczenia sieci kontynuuje się ten proces obliczając koszt $J(h)$. Propagacja wsteczna (*back-propagation*) pozwala na przepływ informacji od funkcji kosztu wstecz sieci neuronowej, aby ostatecznie obliczyć gradient. Zasada działania algorytmu propagacji wstecznej błędu polega na sukcesywnym aktualizowaniu wag i biasów oraz przesyłaniu wstecz po warstwach sieci. Dzięki temu jesteśmy wstanie wyuczyć sieć oraz obliczyć optymalne wagi i biasy dla całej sieci neuronowej.

[5] Deep Learning techniques for music geneation - A survay s.63

1.5.2. Autoencodery

Autoencoder jest szczególnym przypadkiem sieci neuronowej. Posiada jedną warstwę ukrytą, a rozmiar wartwy wejściowej musi być równy rozmiarowi wartwy wyjściowej, tworząc w ten sposób symetryczną sieć, której kształt przypomina klepsydrę. Przykład autoencodera przedstawiono na rysunku 1.8.



Rysunek 1.7: Przykład modelu autoencodera

Podczas uczenia autoencodera przedstawia się dane wejściowe jako cel. W ten sposób ta architektura stara się odtworzyć funckje identyczności. Zadanie nie jest trywialne jak mogło by się zdawać, ponieważ zazwyczaj ukryta warstwa jest mniejszego rozmiaru niż dane wejściowe. Z tego względu autoencoder jest zmuszony do wydobycia istotnych cech danych wejściowych, skompresowania, a następnie jak najwierniejszego ich odtworzenia. Część kompresująca dane nazywana jest encoderem, natomiast część dekompresująca decoderem. Wektor cech, które zostały odkryte przez autoencoder nazywane są zmiennymi utajnionymi (*latent variables*). Zarówno encoder jak i dekoder można wyodrębnić z autoencodera i wykorzystywać go jako osobną sieć neuronową.

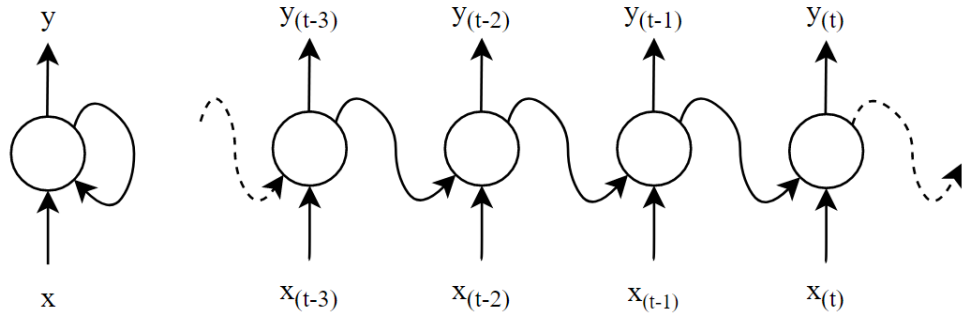
Ciekawą cechą decodera jest jego generatywny charakter, ponieważ dostarczając zupełnie nowe informacje jako zmienne wejściowe, decoder odtworzy je na podobieństwo danych, na których został nauczony.

1.5.3. Rekurencyjne sieci neuronowe

Rekurencyjne sieci neuronowe (*recurrent neural networks - RNN*) w uproszczeniu są to MLP posiadające pamięć. Wykorzystywane są do analizawania i

przewidywania sekwencji wartości uporządkowanych w czasie. Rekurencyjnych sieci neuronowe znalazły zastosowanie w przetwarzaniu języka natutralnego, np. tłumaczenia na różne języki świata. Potrafią poradziac sobie z różnej długości sekwencjami od krótkich zawierających kilka elementów do bardzo długich jak próbki audio, czy tekst zawierający dziesiątki tysięcy kroków czasu.

Rekurencyjne sieci neuronowe działają podobnie do omawianych w sekcji 1.5.1 sieci jednokierunkowych z tym wyjątkiem, że kierunek przepływu informacji płynie również wstecz sieci. Jeden neuron sieci RNN otrzymuje dane wejściowe $x_{(t)}$, wytwarza dane wyjściowe $y_{(t)}$, a następnie wysyła te dane wyjściowe z powrotem do samego siebie. W ten sposób neuron RNN posiada dwa wejścia $x_{(t)}$ oraz $y_{(t-1)}$. Możemy również zaprezentować sieć RNN w postaci udwiniętej w czasie (*unrolled through time*).



Rysunek 1.8: Rekurencyjny neuron (po lewej) odwinięty w czasie (po prawej)

Gdyby rozważyć całą warstwę neuronów tego typu, wtedy wartswa przyjmowała by dwie macierze wag W_x oraz W_y. Dane wyjściowe całej warstwy zostaną obliczone wtedy zgodnie ze wzorem

$$y_{(t)} = AF(W_x^\top x_{(t)} + W_y^\top y_{(t-1)} + b)$$

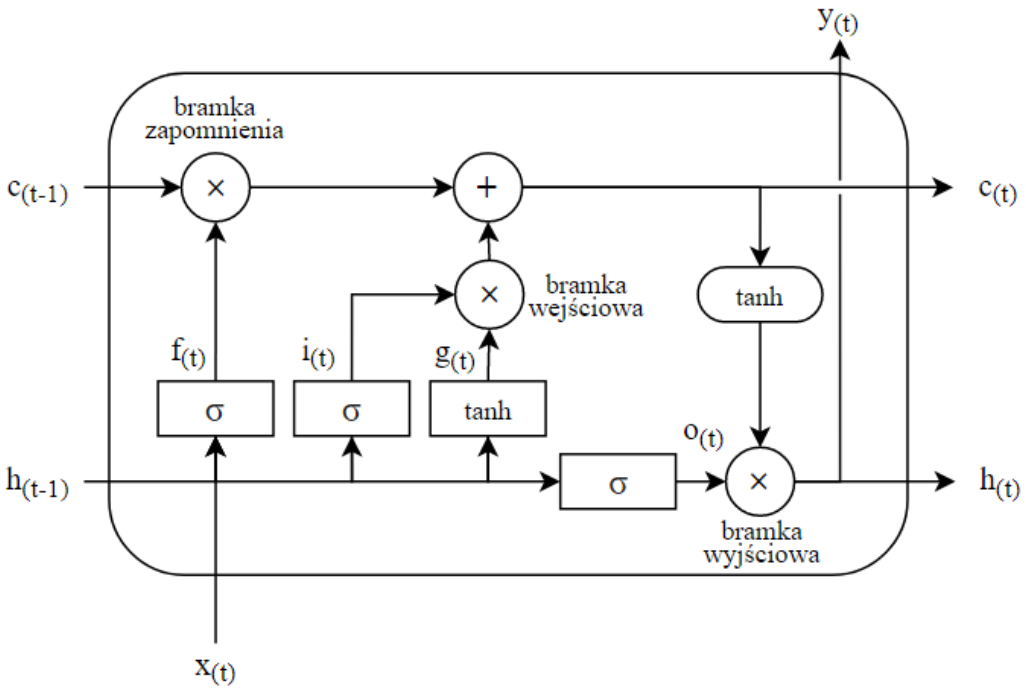
Aby wytrenować sieć neuronową stosuje się propagację wsteczną w czasie (*backpropagation through time - BPTT*). Polega ona na odwinięciu sieci RNN, a następnie zastosowania zwykłej metody wstecznej propagacji. [6]

1.5.4. LSTM

Komórki LSTM (*long-short term memory*) są rozszerzeniem neuronów sieci rekurencyjnych. Pozwalają wykrywać zależności w danych w długim okresie.

[6] Hands-on machine learning with scikit-learn, keras and TensorFlow s.497

Posiadają dwa wektory opisujące stan neuronu. Wektor $h_{(t)}$ określa stan krótkookresowy i wektor $c_{(t)}$ określa stan długookresowy.



Rysunek 1.9: Komórka LSTM

Główny pomysł na funcjonowanie komórek LSTM był taki, aby sieć sama mogła się nauczyć jakie informacje są istotne i je przechować, a które informacje można pominąć, zapomnieć. Schemat komórki LSTM przedstawiono na rysunku 1.9. Aby to osiągnąć powstała idea bramek (*gates*), oraz kontrolerów bramek (*gate controllers*). W komórce LSTM wyróżniamy trzy bramki. Bramkę zapomnienia (*forget gate*) sterowaną przez $f_{(t)}$, bramkę wejściową(*input gate*) sterowaną przez $i_{(t)}$, oraz bramkę wyjściową (*output gate*), sterowaną przez $o_{(t)}$. Przepływ danych w komórce LSTM zaczyna w miejscu gdzie wektor wejściowy $x_{(t)}$ i poprzedni którkoterminowy stan $h_{(t-1)}$ trafiają do czterech warstw. Główną warstwą jest ta zwracająca $g_{(t)}$. W podstawowej komórce RNN jest tylko ta warstwa. Pozostałe trzy wartswy po przejściu przez funkcje logistyczne trafiają do bramek. Bramka zapomnienia kontroluje, które informacje z długookresowego stanu $c_{(t-1)}$ powinny zostać wykasowane. Bramka wejściowa kontroluje jakie informacje z $g_{(t)}$ powinny zostać przekazane dalej i dodane do nastepnego stanu długookresowego $c_{(t)}$. Bramka wyjściowa odpowiada za wybranie odpowiednich elementów z stanu długookresowego i przekazanie ich

następnych kroku. Wynik komórki zostaje przekazany do wyjścia komórki $y_{(t)}$ oraz jako następny stan krótkoterminowy $h_{(t)}$.

Kolejne etapy komórki LSTM obliczane są zgodnie z poniższymy wzorami:

$$
\begin{aligned}
i_{(t)} &= \sigma(W_{xi}^{\top} x_{(t)} + W_{hi}^{\top} h_{(t-1)} + b_i) \\
f_{(t)} &= \sigma(W_{xf}^{\top} x_{(t)} + W_{hf}^{\top} h_{(t-1)} + b_f) \\
o_{(t)} &= \sigma(W_{xo}^{\top} x_{(t)} + W_{ho}^{\top} h_{(t-1)} + b_o) \\
g_{(t)} &= \sigma(W_{xg}^{\top} x_{(t)} + W_{hg}^{\top} h_{(t-1)} + b_g) \\
c_{(t)} &= f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \\
y_{(t)} &= h_{(t)} = o_{(t)} \otimes tanh\left(c_{(t)}\right),
\end{aligned}
$$

gdzie W_{xi}, W_{xf}, W_{xo}, W_{xg} są to macierze wag dla każdej w czterech warstw połączonych z wektorem wejściowym $x_{(t)}$, W_{hi}, W_{hf}, W_{ho}, W_{hg} są to macierze wag dla każdej w czterech warstw połączonych z poprzednim krótkookresowym stanem $h_{(t-1)}$ a b_i, b_f, b_o, $b_g$to biasy dla każdej z tych warstw. [7]

Funckja *tanh* to tangens hiperboliczny, jedna z funkcji sigmoidalnych. To co różni funkcję *tanh* od σ to zakres przyjmowanych wartości. Tangens hiperboliczny przyjmuje wartości z przedziału $(-1, 1)$

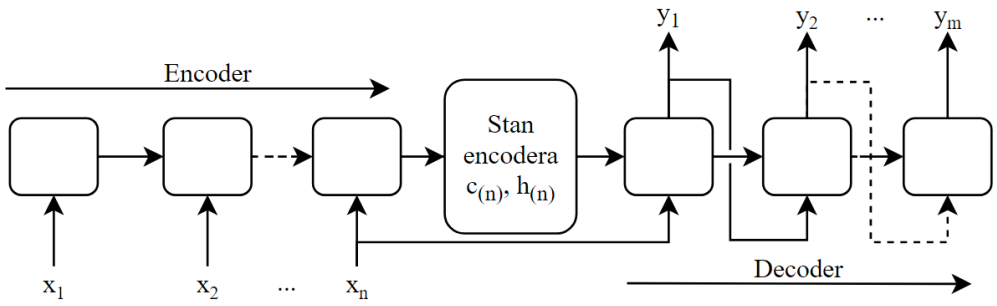
1.5.5. Sequance-to-sequance

Model w architekturze Sequence-to-sequance (*seq2seq*) został wynaleziony z myślą o tłumaczeniu maszynowym języków, ale zastosowanie dla niego znaleziono również w rozpoznawaniu mowy, opisywaniu wideo, czy tworzeniu chatbotów. Jego główną zaletą jest przetwarzanie sekwencji elementów o różnych długościach. Jest to naturalne, ponieważ tłumacząc z języka na język często tą samą sentencję można wyrazić różną liczbę słów. Dla przykładu zdanie po Polsku "Co dzisiaj robisz?" zawiera trzy słowa, natomiast przetłumaczone na Angielski "What are you doing today?" zawiera pięć słów. Nie można tego osiągnąć zwykłą siecią LSTM, dlatego model seq2seq został zaprojektowany, aby móc go zastosować do tego typu problemów. [8]

Model sequence to sequance ma dwie części, encoder i decoder. Obie czeście są w zasadzie dwiema zupełnie innymi modelami, połączonymi ze sobą w jedną wielką sieć. Zadaniem encodera, podobnie jak zostało to opisane w rozdziale 1.5.2 o autoencoderze, jest wydobycie z wektora wejściowego najistotniejszych informacji i skompresowanie ich. Następnie wektor stanu encodera jest przekazywany do decodera, który na jego podstawie rekonstruuje sekwencję.

[7] Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow s.517

[8] https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346 26.05.2020 14:58



Rysunek 1.10: Architektura modelu sequance-to-sequance

Więcej szczegółów technicznych dotyczących modelu sequance-to-sequance przedstawię w dalszych rozdziałach pracy.

Reprezentacja danych muzycznych

W tym rozdziale przedstawię podstawowe koncepcje muzyczne, sposoby reprezentacji muzyki oraz omówię podstawy działania protokołu MIDI.

2.1. Podstawowe koncepcje

2.1.1. Dźwięk muzyczny

Drgania powietrza z otoczenia człowieka są przetwarzane w mózgu i rozumiane jako dźwięki. Takie drgania nazywamy falą dźwiękową. Dzwięk muzyczny jest to fala dźwiękowa, którą wytwarza intrument muzyczny. Dzwięk muzyczny charakteryzuje się trzema podstawowymi parametrami:

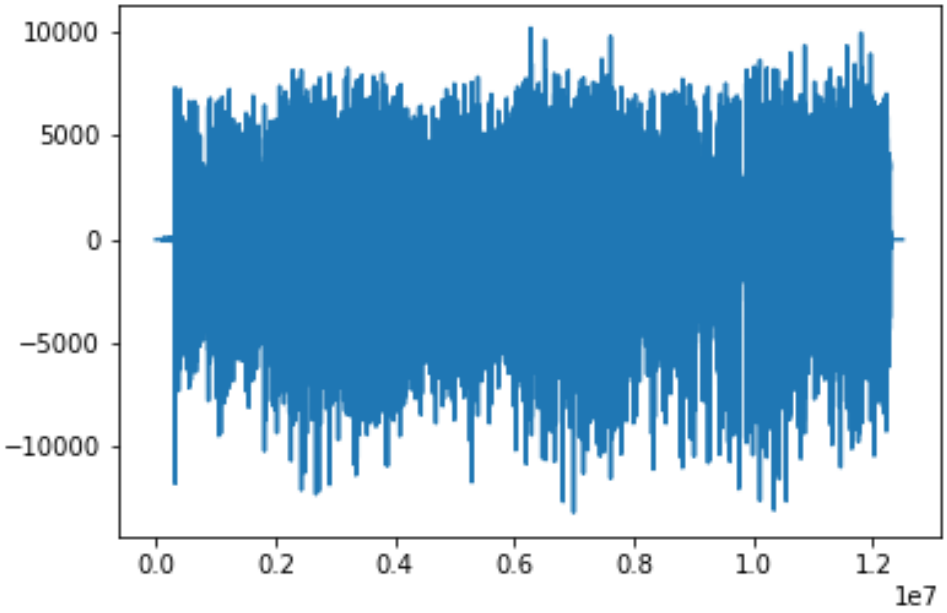
— wysokość (ang. pitch) - jest to częstotliwość drgań wyrażona w hercach. Im większa częstotliwość tym dźięk jest rozumiany jako wyższy, Zakres słyszalny dla człowieka wynosi od 20Hz do 20kHz.
— głośność (ang. velocity) - jest to amplituda drgań fali dźwiękowej. Im większa aplituda, tym dźwięk jest odczuwany jako głosniejszy,
— długość (ang. duration) - jest to czas z jakim dźwięk wybrzmiewa, np. 2 sekundy.

2.1.2. Sygnał dźwiękowy

W rzeczywistości, utwór muzyczny jest zazwyczaj kombinacją wielu fal dźwiękowych, o różnych charaketystykach i nazwywany jest sygnałem dźwiękowym. Wizualizację sygnalu dźwiękowego przedstawiono na Rysunku 2.2

2.1.3. Zapis nutowy

Reprezentacja muzyki jako sygnału dźwiękowego przechowuje informacje o dokładnym brzmieniu danego utworu tzn. jakie drgania należy wytworzyć, aby móc odtwozyć dźwięki. Taki zapis nie informuje nas bezpośrednio jakie instrumenty zostały użyte, jakie wysokości i długości dźwięków zostały wykorzystane. Dlatego ludzkość na przestrzeni wieków opracowała abstrakcyjne objekty, które reprezentują utwór w czytelny dla człowieka sposób.



Rysunek 2.1: Przykład przebiegu fali dźwiękowej

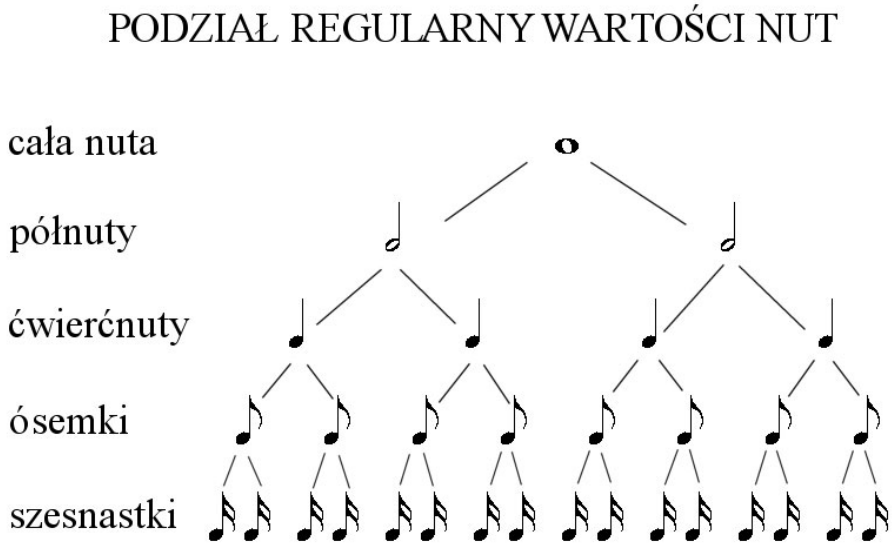
Tempo

W muzyce symbolicznej tempo informuje nas o prędkości utworu. W muzyce klasycznej stosowało się opisowy sposób dostosowywania tempa np. Allegro - Szybko lub Adagio - wolno. Jak można szybko stwierdzić są to zwroty subiektywne i nie wyznaczają tempa jednoznacznie. Obecnie wyraża się tempo w liczbie uderzeń na minutę (BPM ang. beats per minute). I tak Allegro jest to od 120 do 168 BPM a Adagio od 66 do 76 BPM. [1]

Nuta

Nuta jest to graficzna reprezentacja dźwięku muzycznego. Informuje nas ona o dwóch parametrach dźwięku, wysokości oraz długości dźwięku. Długość dźwięku nazywa się jej wartością. Podstawową wartością nuty jest ćwierćnuta, odpowiada ona jednemu uderzeniu. Dla przykładu w tempie 60 BPM w ciągu minuty zagramy dokładnie 60 ćwierć nut. Kolejne wartości tworzone poprzez sumowanie lub podział długości ćwierćnuty. Półnuta trwa tyle co dwie ćwierćnuty, cała nuta tyle co dwie półnuty, ósemka trwa połowę czasu ćwiercnuty, a szesnastka połowę ósemki itd.

[1] źródlo: http://www.classicalmusiccity.com/search/article.php?vars=446/Basic-Tempo-Markings.html 5 kwietnia 19:37



Rysunek 2.2: źródlo: https://www.infomusic.pl/poradnik/46934,poradnik-teoria-muzyki-rytm 5 kwietnia 2020 12:46

Tak jak pisałem wcześniej, wysokość dźwięku jest to częstotliwość drgań fali dźwiękowej wyrażona w hercach. W muzyce symbolicznej dla uproszczenia wybrane częstotliwości zostały nazwane literami alfabetu C, D, E, F, G, A, H. Każdej literze przypisana jest częstotliwość zgodnie z Tabelą 2.1

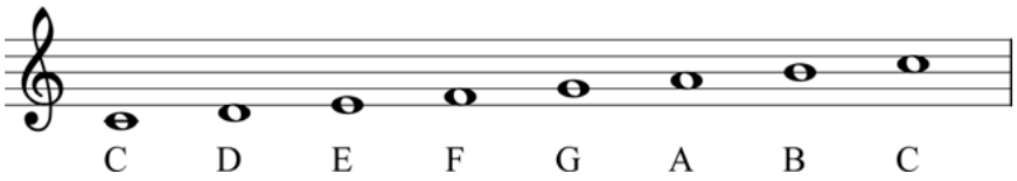
Dzwięk	Częstotliwość
C_4	261,6
D_4	293,7
E_4	329,6
F_4	349,2
G_4	391,9
A_4	440,0
H_4	493,9

Tabela 2.1: Dźwięki symboliczne oraz ich częstosliwości

W zapisie nutowym aby nucie nadać wysokość, umieszcza się ją w odpowiednim miejscu na pięciolonii. Przedstawione powyżej dźwięki zapisalibyśmy w taki sposób jak przedstawiono na Rysunku 2.3

Oktawy

Oktawą nazywamy zestaw ośmiu nut od C do H. Podane w Tabeli 2.1 częstotliwości nut odpowiadają dźwiękom w oktawie czwartej. Dlatego w indeksie



Rysunek 2.3: źródlo: https://amplitudaschool.weebly.com/lekcja-11.html 5 kwietnia 2020 13:24

dolnym nuty widnieje liczba 4. Aby utworzyć dźwięk, np. A_5 należy pomnożyć częstotliwość dźwięku A_4 razy dwa, natomiast aby utworzyć dźwięk A_3, należy tę częstotliwość podzielić przez dwa.

$$A_5 = 440Hz * 2 = 880Hz$$

$$A_3 = 440Hz/2 = 220Hz$$

W ten sposób możemy utworzyć nieskończenie wiele oktaw, jednak w rzeczywistości używa się nut od C0 do C8.

Akord

Gdy w jednym momencie zabrzmią dwie lub więcej różnych nut, wtedy mówimy o akordzie. Akord potrafi dodać emocje do brzmienia całego utworu.

2.2. Cyfrowa reprezentacja muzyki symbolicznej

2.2.1. Standard MIDI

Standrd MIDI (ang. Musical Instrument Digital Interface) został stworzony w 1983 aby umożliwić synchronizację i wymianę informacji między elektronicznymi urządzeniami muzycznymi takimi jak syntezatory, keyboardy czy sekwencery. W późniejszych latach został on zaadaptowany do środowiska komputerowego jako cyfrowa reprezentacja muzyki symbolicznej.

Wiadomości

Plik MIDI zawiera zestaw wiadomości przesyłanych w czasie rzeczywistym o każdej nucie w utworze. Dwie wiadomości, które są dla nas szczególnie istotne to:

— note_on, który sygnalizuje aby rozpocząć grać nutę,
— note_off, który sygnalizuje aby zakończyć grać nutę.

Dla przykładu wiadomość:

```
note_on channel=0 note=48 velocity=100 time=0
note_on channel=0 note=53 velocity=100 time=0
note_on channel=0 note=60 velocity=100 time=0
note_on channel=0 note=48 velocity=0 time=220
note_on channel=0 note=48 velocity=100 time=0
note_on channel=0 note=53 velocity=0 time=0
note_on channel=0 note=55 velocity=100 time=0
note_on channel=0 note=60 velocity=0 time=0
```

Rysunek 2.4: Fragment protokołu MIDI

note_on channel 0 note 48 velocity 100 time 0

oznacza aby na kanele 0 zagrać dźwięk nr 48 z głośnością 100 w momencie 0 utworu. Nie informuje nas on jednak o długości trwania dźwęku. Aby zakończyć dźwęk, należy wysłac wiadomość:

note_off, channel 0, note 48, velocity 100, time 24.

Zwróćmy uwagę że aby ustalić wartość nuty, potrzebujemy odebrać dwie wiadomości. Różnica między parametrami time, informuje nas o długości nuty. W tym przypadku jest to 24. Co oznacza ćwierćnutę.

Rozdzielczość

Czas w MIDI jest reprezentowany jako liczba naturalna i jest on zależny od ustalonego tempa utworu. Standardowa rozdzielczość pliku MIDI to 24. Oznacza to, że jedna jednostka czasu odpowiada jednej dwudziestejczwartej jednego udeżenia.

Kanały

Plik MIDI posiada 16 kanałów numerowanych od 0 do 15. Każdy kanał odpowiada intrumentowi lub ścieżce. Kanał 9 jest kanałem zarezerwowanym na intrumenty perkusyjne.

Nuty

Nuty w formacie MIDI opisane są kolejnymi cyframi naturalnymi w przedziale od 0 do 127. Odpowiada to dźwękom od C_0 do C_8. Dla przykładu nuta 69 odpowiada A_4, a nuta 47 odpowiada B_2.

Wyjątkiem są nuty z kanału dziewiątego, gdzie istnieją tylko nuty z zakresu od 35 do 81 i każda nuta odpowiada innemu elementowi perkusyjnemu np. 35 to stopa, a 37 to werbel.

Głośność

Za głośność dźwęku odpowiada parametr velocity, który jest liczbą z przedziału od 0 do 127. Im większa jest wartość tym głośniej wybrzmi dźwęk.

Program

Program w kontekscie standardu MIDI oznacza intrument który ma zagrać nuty. W standardzie GM (ang. General MIDI), jest 16 grup intrumenów a w każdej z nich znajduje się po 8 intrumentów. Są to piania, chromatyczne perkusje, organy, gitary, basy, intrumenty smyczkowe, zestawy intrumentów, intrumenty dmuchane blaszane, intrumenty dmuchane drewniane, flety, syntezatory prowadzące, syntezatory uzupełniające, efekty syntetyczne, instrumenty etniczne, perkusjonalia i efekty dzwiekowe.

Ścieżka

Ścieżka (ang. Track) grupuje nuty aby podzielić utwór muzyczny na różne intrumenty. Protokół MIDI pozwala aby grać wiele ścieżek dzwiękowych jednocześnie, wtedy mówimy o muzyce polifonicznej lub multiintrumentalnej.

Rozdział 3

Projekt

W tym rozdzialę opiszę w jaki sposób zbudowałem swój własny geneator muzyki, jak przechodził procesz uczenia, jakie próbki udało mi się wygenrować. Opis kodu który napisałem.

Rozdział 4

Podsumowanie

Ostateczne wnioski, czy muzyka generowana komputerowa da się lubić? Czy to pozytywnie wpłynie na przemysł muzyczny? Tak i nie. Może złużyć jako inspiracja dla muzyków, proces wspierający. Z drugiej strony może obnizy koszty produkowania muzyki pop, która i tak jest już bardzo powtarzalna. Czy sieci neuronowe nauczą się produkować Hity?

Bibliografia

[1] Briot, J.P., Hadjeres, G., Pachet, F.D. (2019): *Deep Learning Techniques for Music Generation - A Survey. arXiv:1709.01620v3*

[2] Goodfellow, I., Bengio, Y., Courville, A. (2016): *Deep Learning. MIT Press.*

[3] Zocca, V., Spacagna, G., Slater, D., Roelants, P. (2018): *Deep Learning. Uczenie głębokie z językiem Python. Helion.*