VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ SKLÁDÁNÍ KLASICKÉ HUDBY

AUTOMATIC COMPOSITION OF CLASSICAL MUSIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MAREK MAJER**
AUTHOR

VEDOUCÍ PRÁCE **Ing. KAREL BENEŠ**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Majer Marek**
Obor: Informační technologie
Téma: **Automatické skládaní klasické hudby**
Automatic Composition of Classical Music
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se rekurentními neuronovými sítěmi jako nástrojem pro modelování sekvencí
2. Na volně dostupných datech natrénujte model klasické hudby
3. Experimentujte s postupy vzorkování hudby z tohoto modelu
4. Ověřte kvalitu takto generované hudby

Literatura:

- podle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Beneš Karel, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce pojednává o používání rekurentních neuronových sítí pro vytváření klasické klavírní hudby. Jsou zde popsány jednotlivé možnosti nastavení modelu, způsob práce s daty a výsledky získané ze studia rekurentních neuronových sítí

Abstract

This document describes using recurrent neural networks for generating clasicial piano music. It also mentions various settings for model, how to work with data and the results from studying recurrent neural networks

Klíčová slova

Rekurentní neuronové sítě, generování hudby, piáno

Keywords

Recurrent neural networks, music generation, piano

Citace

MAJER, Marek. *Automatické skládání klasické hudby.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Karel Beneš

Automatické skládání klasické hudby

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana inženýra Karla Beneše. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Marek Majer
16. května 2017

Poděkování

Chtěl bych poděkovat za odborné vedení mé bakalářské práce panu inženýru Karlu Beneši.

Obsah

Kapitola 1

Úvod

Hudba je součástí každodenního života. Spousta lidí ale hudbu většinu života jenom pasivně přijímá a když přijde doba, kdy by se rádi naučili hrát na nějaký hudební nástroj, je pro ně těžké vytvořit vlastní skladby. Pro naučení hraní na nástroj už existuje na internetu spousta rad a návodů, zároveň je na internetu spousta informací a notových zápisů pro známé skladby.

Proces psaní nových skladeb ale nelze jednoduše vysvětlit. Pokud má člověk pár tónů, které mu hezky znějí dohromady, je pro něj většinou problém vytvořit zbytek skladby, zvláště v případě, pokud teprve začíná hrát na daný nástroj. Každá skladba se od ostatních v něčem liší, obsahuje svůj rytmus a opakující se motiv, který je pro danou skladbu typický a nelze si tedy vyhledat jaký tón se má hrát po konktrétním tónu, je potřeba se na skladbu dívat jako na celek.

Hudbu si lze představit jako spoustu tónů chronologicky seřazených za sebou, tvořící hudební sekvenci. K modelování hudebních sekvencích, ve kterých silně závisý na čase ve skladbě, se zdají být velice vhodné neuronové sítě [2]. Neuronové sítě jsou schopné naučit se spoustu informací a následně díky tomuto učení lze docílit odhadu, který tón by byl nejčastěji zahrán, i v případě, že se ve skladbě objeví nějaký nečekaný tón. Neuronové sítě pak budou schopny reagovat na tenhle nový tón v souladu s tím, co se dříve naučili.

Hlavní náplní této práce je otestování jednotlivých přístupů k neuronovým sítím na hudebních datech.

K demonstrování možností neuronových sítí, jsem vytvořil jednoduchou aplikaci. Neuronová síť bude srdcem celé aplikace, uživatel vytvoří část svoji skladbu, která bude celá odeslána neuronové síti a ta za něj zvolí notu v následujícím čase.V případě potřeby lze neuronové síti zadat pouze první notu, pak bude možné rekurentně volat neuronovou síť a vytvořit celou skladbu.

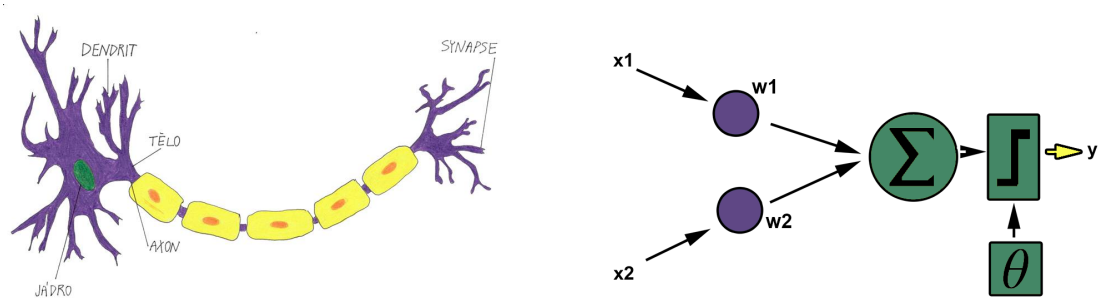
Kapitola 2

Neuronové sítě

Neuronová síť (Neural network - NN) je model zpracování informací, který byl inspirován biologickým nervovým systémem (viz obrázek 2.1).

Tělo biologického neuronu sbírá signály ze svých dentritů, agreaguje je a dále vysílá signály přes dlouhou, tenkou trubici nazývanou axon, která se dále rozděluje na spoustu dalších větví. Na konci každé větve je struktura nazývaná synapse, která přeměňuje aktivitu z axonu na elektrický efekt, který tlumí nebo vyvolává aktivitu v připojených neuronech.

Protože jsou lidské znalosti o neuronech neúplné a výpočetní síla limitovaná, jsou umělé modely reprezentovaný jako ideální verze skutečných neuronů. Umělá neuronová síť se skládá z velkého počtu propojených neuronů, které pracují jako celek na vyřešení specifického problému. Neuronové sítě slouží k modelování určité funkce, u které známe pouze její vstupy a výstupy.



(a) Neuron v lidském mozku

(b) Neuronu v počítači

Obrázek 2.1: Srovnání biologického a umělého neuronu. Fialové vstupy neuronu sbírají informace, se kterými se pracuje v zeleném jádru neuronu. Informace jsou dále přes žlutý výstup poslílány do dalších neuronů.

2.1 Architektura neuronových sítí

Výpočetní schopnosti jednoho neuronu nejsou pro modelování složitějších funkcí dostačující, proto jsou neurony propojovány do sítí. Množina neuronů, která přijímá signál z jiné množiny neuronů, případně okolí, se nazývá vrstva. Signál proudí z první vrstvy vstupní

přes skryté vrstvy až do poslední vrstvy výstupní. Vstupní a výstupní vrstva má většinou jiný počet neuronů, než vrstvy skryté. Skryté vrstvy jsou určené podle úkolu neuronové sítě, zatímco skryté vrstvy mohou mít jakkoukoliv šířku.

2.1.1 Umělý neuron

První model neuronu byl navržen v roce 1943 [9]. Tento neuron se skládá ze tří částí: synaptických vah (w_i), biasu (θ) a aktivační funkce (F). Vstupem neuronu je vektor $\vec{x}$ o m prvcích, kde m je počet neuronů připojených do daného neuronu, v případě vstupní vrstvy se jedná o počet vstupů z okolí. Váhy mezi neurony (w_i) mohou nabývat kladné i záporné hodnoty. Mohou tedy modelovat jak aktivační, tak inhibiční synapse.

Výstupem samotného neuronu je jedna hodnota, neuron tedy provádí transformaci z vektoru čísel na jedno číslo. K váženému součtu jednotlivých vstupů je připočten bias a výsledek je použit jako vstup aktivační funkce. Výstupem aktivační funkce je potom výstup samotného neuronu:

$$y = F(\sum_{i=1}^{m} w_i x_i + \theta) \tag{2.1}$$

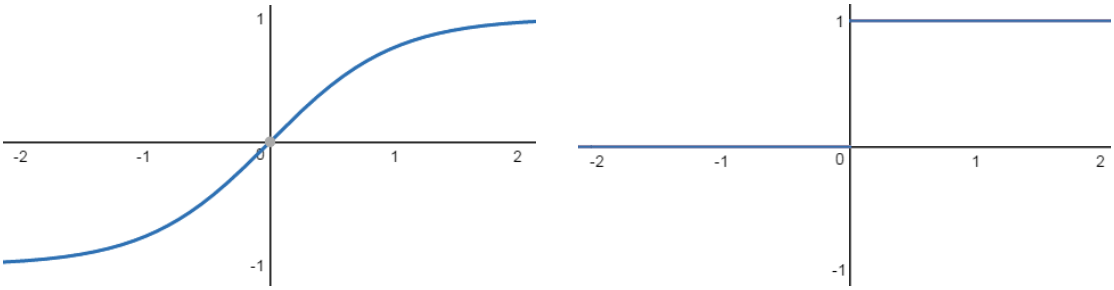
2.1.2 Aktivační funkce

Neuronové sítě musí být schopné aproximovat požadovanou funkci J, proto je potřeba zajistit nelinearitu. Bez nelineární aktivační funkce by se NN chovala jako jedna vrstva neuronů, která by byla pouhým součinem jednotlivých vrstev. Aktivační funkce (viz obrázek 2.2) by měla být jednodušše derivovatelná, aby bylo možné jednoduše vypočítat její gradient k dalšímu učení NN. Funkci se dává jako vstup součet váženého vstupu a biasu, díky aktivační funkci je také možné stlačit lineární součet vah do určitého intervalu.

U původního neuronu, byla používána kroková funkce. Výsledek této funkce je závislý na prahu. Všechny vstupy, které jsou menší než daný prah, nabývá hodnoty 0, naopak hodnoty vyšší nabývají hodnoty 1.

Logická sigmoida patří k nejčastěji používaným aktivačním funkcím. Výstupem funkce logická sigmoida je reálné číslo z intervalu $(0, 1)$:

$$o(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$



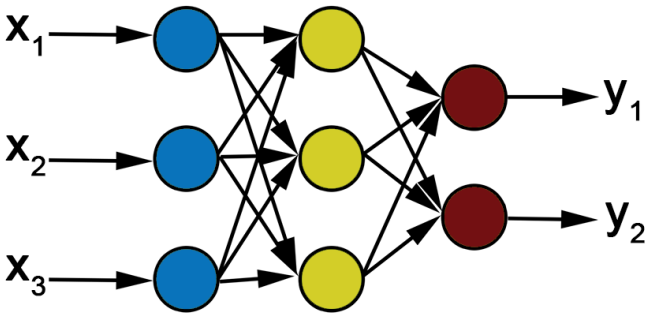
(a) Logická sigmoida (b) Kroková funkce

Obrázek 2.2: Nelineární aktivační funkce používané v umělých neuronových sítí, Log. sigmoida (a) je vhodnější pro komplexnější neuronové sítě, protože přidává prvek nejistoty, zatímco kroková funkce (b) umožňuje pouze přepínat mezi 0 a 1.

2.1.3 Dopředná neuronová síť

Dopředná neuronová síť (Feedforward neural network - FFNN) je kolekce neuronů seskupených do vrstev a propojených s každou vedlejší vrstvou (viz obrázek 2.3). Signál postupně prochází neuronovou síť od vstupní vrstvy až k výstupní.

Vstupní vrstva je vrstva, které jsou posílány signály z okolí. Vrstvy, které nemají přímé spojení s okolím, jsou nazývány skryté. Počet skrytých vrstev není omezen. Výstupní vrstva je vrstva, ve které se nachází výsledek neuronové sítě.



Obrázek 2.3: Dopředná neuronová síť, na vstupní (modrou) vrstvu je síti přiveden vstup z okolí, neurony vrstev si posílají signál přes libovolný počet skrytých (žlutých) vrstev, na konci lze získat výstup z výstupní (červené) vrstvy.

2.2 Trénování neuronových sítí

Učení v biologických systémech funguje na principu úpravy synaptických spojů mezi neurony. Na stejném principu funguje i učení umělých neuronových sítí. Neuronové sítě jsou

trénovány příkladem, kdy je síti poskytován vstup a odpovídající výstup. Síť má na začátku nastavné náhodné váhy a pomocí optimalizačního algoritmu jsou jednotlivé váhy upravovány, tak aby byl výstup bližší vstupním datům.

2.2.1 Chyba neuronové sítě

Aby bylo možné upravit hodnoty vah, je potřeba určit chybu výstupu NN, to je rozdíl požadovaného výstupu a výstupu který vytvořila NN. V této práci výstupem není pouze hodnota jednoho výstupního neuronu, ale více výstupních neuronů může mít výstup v 1 nezávisle na ostatních. Chyba se vypočítá pomocí cross-entropy. Kde $X = \left\{x^{(1)}, \cdots, x^{(n)}\right\}$ je množina vstupních dat, $Y = \left\{y^{(1)}, \cdots, y^{(n)}\right\}$ je množina odpovídajících výstupů a $a(x)$ je výstup neuronové sítě při vstupu x:

$$E(X,Y) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} \ln a(x^{(i)}) + \left(1 - y^{(i)}\right) \ln\left(1 - a\left(x^{(i)}\right)\right)\right] \tag{2.3}$$

2.2.2 Gradient descent

Pro zlepšení neuronové sítě je potřeba nalézt parametry, které povedou k největšímu snížení chyby. K tomu slouží optimalizační algoritmus gradient descent (GD). Gradient je vektor, který ukazuje na nejvyšší růst hodnot ve funkci. GD je způsob aktualizace parametrů $\vec{\theta}$ aproximační funkce $J(\vec{\theta})$ v opačném směru gradientu dané funkce.

Pro použití GD je potřeba znát derivaci chyby, derivaci váhy neuronu a zvolit krok učení. Krok učení η u GD udává velikost posunu v opačném směru gradientu. Zvolením nízkého kroku může učení trvat dlouho, naopak zvolením příliš vysokého kroku může zapříčinit kolísání okolo minima, případně i divergenci od minima.

Protože chyba E je vypočítána pouze složením funkcí v neuronech, je to funkce spojitá a diferencovatelná. Gradient chyby se vypočítá jako závislost celkové chyby na jednotlivých vaháh:

$$\nabla E = \left(\frac{\delta E}{\delta w_1}, \frac{\delta E}{\delta w_2}, \cdots, \frac{\delta E}{\delta w_l}\right) \tag{2.4}$$

Batch gradient descent (BGD) musí vypočítat gradient pro celý dataset, aby provedl jednu aktualizaci. Může být tedy velmi pomalý:

$$\theta \leftarrow \theta - \eta * \nabla_\theta J(\vec{x}) \tag{2.5}$$

Stochastic gradient descent(SGD) narozdíl od BGD provádí aktualizaci pro každý z trénovaných příkladů $x^{(i)}$ a odpovídajících výsledků $y^{(i)}$:

$$\theta \leftarrow \theta - \eta * \nabla_\theta J(\theta; \vec{x^{(i)}}, \vec{y^{(i)}}) \tag{2.6}$$

BGD provádí redundantní operace pro velké datasety, protože počítá gradient pro podobné příklady před každou aktualizací parametru. BGD vždy konverguje k minimu funkce, narozdíl od něj SGD může divergovat, to umožňuje naleznutí potencionálně lepších minim, na druhou stranu nedochází k přesné konvergenci k minimu. Toto chování lze vylepšit zvolením nízkého kroku učení.

Mini-batch gradient descent (mSGD) je upravenou verzí předchozích způsobů. Provádí aktualizici pro každou malou dávku (batch) n trénovacích dat:

$$\theta \leftarrow \theta - \eta * \nabla_\theta J(\theta; x^{(i:i+n)}, y^{(i:i+n)}) \tag{2.7}$$

Tímto způsobem se snižují rozdíly v úpravě parametrů, díky tomu může docházet ke stabilnější konvergenci a mohou být použity optimalizované způsoby počítání matic. Počítání gradientu v mini-batch je tedy velmi efektivní. Velikost jednotlivých dávek obvykle bývá v rozmezí 16 až 256 vzorků.

2.2.3 Výpočet gradientu

Protože je vypočítání gradientu příliš zdlouhavé, používají se různé optimalizační metody, jednou z nejpoužívanějších metod je backpropagation algoritmus [11]. Algoritmus lze rozdělit na dva kroky, dopředné šíření sekvencí ze vstupu na výstup a následné šíření chyby z výstupu na vstup. $w_{pq}(t)$ je váha synapse z neuronu p do neuronu q v čase t, d_i je cílová hodnota neuronu i, o_i je aktuální hodnota neuronu i. Algoritmus se liší v závislosti jestli se jedná o vrstvu výstupní (neuron i) nebo vrstvu skrytou (neuron j):

$$\begin{aligned} w_{pq}(t+1) &= w_{pq}(t) + \Delta w_{pq} \\ \Delta w_{pq}(t+1) &= \eta * \delta_q * o_p \\ \delta_i &= (d_i - o_i) * o_1 * (1 - o_i) \\ \delta_j &= o_j * (1 - o_j) * \sum w_{ji} * \delta_i \end{aligned} \tag{2.8}$$

2.2.4 RMSprop

Problémem gradient descentu je zvolení vhodného parametru učení. Velikost gradientu se může v průběhu trénování měnit a některé gradienty mohou být příliš velké, některé naopak příliš malé v porovnání s váhy neuronů. Jako optimalizační metoda pro tento problém byl navrhnut RMSprop[4]

Podle znaménka gradientu se adaptivně mění krok učení. Pokud se znaménko nezměnilo zvětšuje se krok učení, pokud se znaménko změnilo krok učení se zmenšuje.

$$\text{KvadratickýPrůměr}(w,t) = 0.9\ \text{KvadratickýPrůměr}(w,t-1) + 0.1(\triangledown E)^2 \tag{2.9}$$

Následné dělení gradientu druhou odmocninou $\text{KvadratickýPrůměr}(w,t)$ zlepšuje další učení. RMSprop je doporučený používat pro rekurentní neuronové sítě.

2.3 Modelování sekvencí

Neuronové sítě jsou výkoné modelovat sekvence a kvůli této vlastnosti, jsou použity v mé práci. Během tréninku neuronové sítě jsou síti předkládány vstupní sekvence a přidružený výstup k těmto sekvencím. Při práci se sekvencemi je podstatné uchovávávat v paměti předchozí prvky ze sekvence, proto pro lepší modelování nestačí pouze dopředné neuronové sítě, ale je potřeba vytvořit určitý druh zpětné vazby.

2.3.1 Rekurentní neuronová síť

Čas je důležitým prvkem v sekvencích a proto je podstatný způsob reprezentace sériově seřazených vstupů. Sítě se zpětnou vazbou nazýváme rekurentní neuronové sítě (RNN).

Způsob používání minulých hodnot byl navrhnut Michaelem I. Jordanem [6], Jordanova neuronová síť funguje jako jednoduchý dynamický systém, ve kterém je předchozí výstup sítě poskytnut jako dodatečný vstup.

V Jordanově práci je stav neuronové sítě v jakémkoliv momentu součet funkce současných vstupů a vstupů v předcházejícím čase. Protože není skrytým jednotkám určena žádná hodnota, mohou se naučit specifický úkol, díky tomu se mohou stát pamětí pro velmi specifické úkoly

Neuronová síť popsaná Elmanem [3] (viz obrázek 2.4) má běžné dopředné spojení ze vstupních neuronů do skrytých neuronů a ze skrytých neuronů do výstupních neuronů. Používá zároveň další neurony, nazývané kontextní jednotky, které umožňují simulovat určitý druh zpětné vazby, rekurenci. Tyto kontextní jednotky poskytují nové vstupy vždy se skrytými jednotkami v předchozí vrstvě a mají lineární aktivační funkci. Díky tomuto způsobu jde zajistit zpětná vazba o jeden časový krok. Elmanova síť lze popsat rovnicemi, ve kterých $\vec{x}_t$ je vstupní vektor, $\vec{h}_t$ je vektor skryté vrstvy, $\vec{y}_t$ je výstupní vektor, W, U jsou váhy jednotlivých vstupů a $\vec{b}$ je bias:

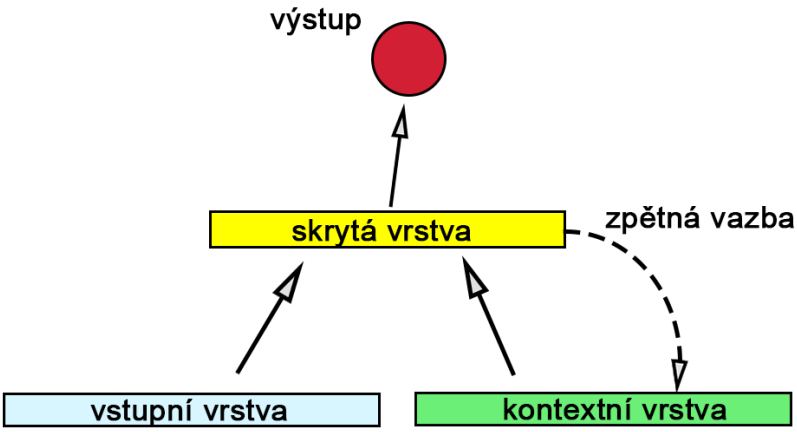
$$\vec{h}_t = \sigma(W_h\vec{x}_t + U_h\vec{h}_{t-1} + \vec{b}_h) \tag{2.10}$$

$$\vec{y}_t = \sigma(W_y\vec{h}_t + \vec{b}_y) \tag{2.11}$$

Obdobným způsobem lze popsat i Jordanovu síť:

$$\vec{h}_t = \sigma(W_h\vec{x}_t + U_h y_{t-1} + \vec{b}_h) \tag{2.12}$$

$$\vec{y}_t = \sigma(W_y\vec{h}_t + \vec{b}_y) \tag{2.13}$$



Obrázek 2.4: Elmanova síť, kontextní vrstva je při prvním průchodu prázdná, při každém dalším průchodu je do skryté vrstvy vždy s výstupem vstupní vrstvy i přidán obsah kontextní jednotky, který je po každém průběhu naplněn starým obsahem skryté vrstvy.

2.3.2 Long Short-Term Memory

Při zpětném šíření chyby v RNN často docházelo buď k prudkému zvýšení chyby, nebo naprostému zmizení chyby. Velikost zpětně šířené chyby exponencionálně záleží na velikosti

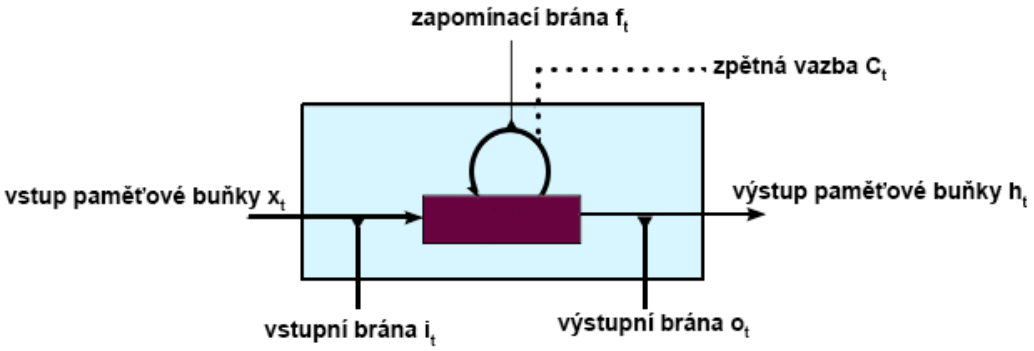
vah. V případě prudkého zvýšení chyby může docházet k oscilaci vah, v případě zmizení chyby bude učení trvat příliš dlouho, nebo nebude fungovat vůbec [5].

Long Short Term Memory (LSTM) sítě jsou speciální verzí rekurentních neuronových sítí, které jsou schopné dlouhodobě udržovat informace v paměti. LSTM řeší problém jednoduchých rekurentních neuronových sítí, které si dokázaly zapamatovat informace jen po velmi krátkou dobu. Narozdíl od toho si LSTM může uchovat informace až po dobu 1000 kroků. [5]

V LSTM se nachází paměťová buňka (viz. obrázek 2.5), která obsahuje jednu centrální jednotku. Navíc se v ní nachází spousta na sebe napojených bran, které vybírají, které informace jsou podstatné pro uchování a způsob jejich uchování. LSTM vrstvy lze pak popsat funkcemi [1]:

$$\vec{i_t} = \sigma(W_i\vec{x}_t + U_i\vec{h}_{t-1} + b_i) \tag{2.14}$$
$$\widetilde{C}_t = tanh(W_c\vec{x}_t + U_c\vec{h}_{t-1} + b_c) \tag{2.15}$$
$$\vec{f_t} = \sigma_g(W_f\vec{x}_t + U_f\vec{h}_{t-1} + b_f) \tag{2.16}$$
$$C_t = \vec{i_t} * \widetilde{C}_t + f_t \otimes C_{t-1} \tag{2.17}$$
$$o_t = \sigma_g(W_o\vec{x}_t + U_o\vec{h}_{t-1} + b_o) \tag{2.18}$$
$$h_t = o_t * tanh(C_t) \tag{2.19}$$

Rovnice 2.14 je vstupní brána v čase t. Rovnice 2.15 odpovídá operaci jednoduché RNN (viz. rovnice 2.10). $\widetilde{C}_t$ je kandidátní hodnota pro stav paměťové buňky v čase t, $\vec{f_t}$ je zapomínací brána paměťové buňky v čase t. Z předchozích hodnot lze vypočítat novou kandidátní hodnotu C_t (viz. rovnice 2.18). Vypočítáním nového stavu paměťové buňky lze vypočítat $\vec{o_t}$ výstupní bránu a následně i samotný výstup h_t.



Obrázek 2.5: Paměťová buňka v LSTM, která slouží k lepšímu uchovávání informace, díky lineární zpětné vazbě a několika bránám, které jsou realizovány jako funkce nad vstupem a vnitřním stavem paměťové buňky. Obrázek inspirován z [1]

Kapitola 3

Data

V následující kapitole jsou popsány základní prvky hudby a způsoby její interpretace v počítačích. Při práci s hudbou v počítačích se užívají dva hlavní formáty uložení dat: zvukový soubor, který je přímo uložená digitální nahrávka zvuku; jeden ze zástupců je například MP3. Druhým způsobem uložení dat je využití předem uznávaného protokolu, podle kterého bude možné v počítači daný zvuk vytvořit, například MIDI formát. Protože se ve formátech typu zvukový soubor mohou nacházet zbytečné zvuky a šumy, používá se v této práci protokolové uložení hudby.

3.1 Hudba

Zvuk je mechanické vlnění hmotných částic. Zdrojem zvuku bývá kmitající těleso, například struna klavíru. Rychlost šíření zvuku závisí na prostředí, ve kterém se daný zvuk šíří. Lidské ucho dokáže vnímat pouze zvuk s frekvencí v rozmezí 16 Hz až 20 kHz.

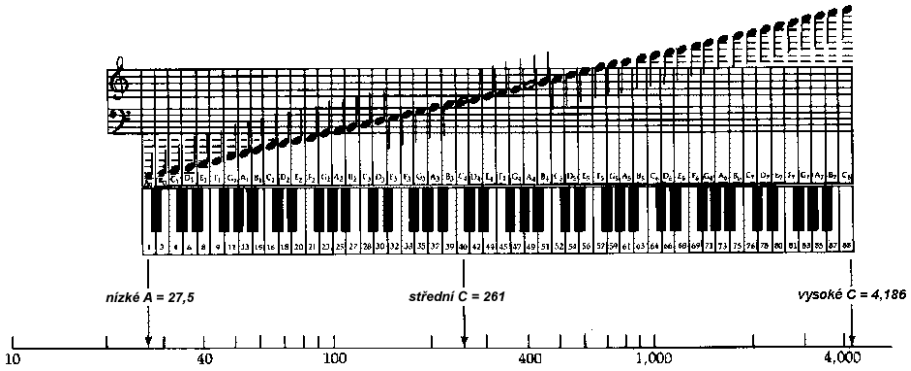
Pro rozlišování jednotlivých tónu v hudbě je hlavní vlastností zvuku jeho frekvence. Tón je charakterizovaný jednou výraznou frekvencí, zatímco šum se skládá z větší oblasti spojitě rozprostřených frekvencí.

Vlastnosti tónu jsou jeho výška (frekvence), jeho délka, síla a barva. V této práci se pracuje pouze s klavírem, zároveň se data budou číst ze zápisu hudby, ve kterém se neudává jeho síla a barva. Síla a barva tónu budou považovány za konstantní. Tóny rozlišujeme podle jejich výšky v rozsahu tónů, se kterými vybraný nástroj pracuje. V případě standardního klavíru se jedná o 88 tónů, v rozsahu od nejhlubšího subkontra "A2"po nejvyšší "c5"(viz obrázek 3.1).

Rytmus je pravidelné střídání přízvučných a nepřízvučných dob. Rytmus lze také definovat jako střídání různě dlouhých not a pomlk. Každá skladba má svůj specifický rytmus.

V jednom okamžiku v čase nemusí hrát pouze jeden tón, správnou volbou současně znějících tónů může docházet k jejich souladu neboli harmonií. Naproti tomu může autor skladby chtít vytvořit u posluchače nepříjemný souzvuk, který nazýváme disharmonií.

Hudbou potom rozumíme organizovaný systém tónů. Podle uspořádání a rytmického členění těchto zvuků můžeme určit kvalitu a estetické působení dané hudby. Uspořádání a členění těchto tónů také určuje styl hudby. I přes to, že se v této práci pracuje pouze s klasickou hudbou, se zde nacházejí určité odlišnosti: Nottingham je souhrn lidové hudba, a proto se v něm nenacházejí takové kombinace tónů, které dokážou vyvolávat úzkost a očekávání jako u chorálu Johanna Sebastiana Bacha (JSBChorales). Zbývající dva datasety



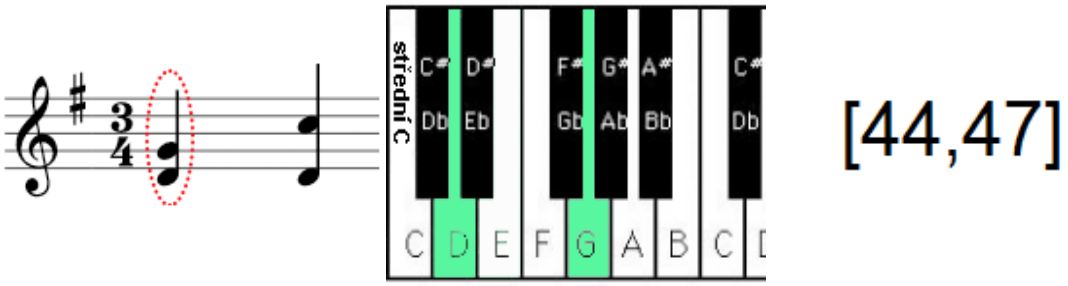
Obrázek 3.1: 88 klapek klasického piána, většina klasických skladeb je hrána okolo středního C.

(MuseData a Piano-midi.de) jsou sbírky skladeb více autorů napsané pro orchestr, proto se v těchto datasetech mohou vyskytovat i tóny jako je nízké A nebo vysoké C.

3.2 Reprezentace hudby a datasety

Piano-roll je způsob ukládání jednotlivých tónů pro samohrající piána. V minulosti byl piano-roll implementován jako páska do které byly vyraženy díry. Tato páska se potom četla čtecí hlavou, která podle vyražení zahraje určitý tón. Tento styl ukládání se kvůli zastaralosti samohrajících pián už moc nepoužívá, je to ovšem vhodný způsob ukládání dat pro další úpravu v počítači.

Jednotlivé tóny jsou uloženy svojí výškou v rozsahu hodnot, u klavíru v této práci je to 88 tónů. Vzorky jsou seřazeny chronologicky za sebou a obsahují informaci o všech tónech hraných v daném čase. Délku tónu určujeme počtem za sebou následujících vzorků. Časová doba jednoho vzorku je předem daná a nejčastěji to je osmina taktu.



(a) (b) (c)

Obrázek 3.2: Ilustrace stejných hudebních dat různými způsoby, (a) je klasická notová osnova s červeně označenými tóny hrajícími ve stejném čase, (b) jsou klávesy, které musí být stisknuty najednou, aby byly stejné noty zahrány na piáno, (c) je reprezentace stejných dat pomocí seznamu.

Pro trénování a testování byly použity 4 hudební datasety. Nottingham obsahující přes 1000 klasických folklórních písniček, JSBChorales obsahující tvorbu Johanna Sebastiana Ba-

cha, MuseData jsou tvořeny kolekcí různých skladatelů tvořících v období let 1690 až 1890. Piano-midi.de je kolekce různých skladatelů, podobně jako MuseData. V této práci se tedy pracuje s rozdílnými skladbami od folklorních písní až po tvorbu nejznámějších skladatelů klasické hudby.

Četnost použitých tónů se v jednotlivých datasetech liší. V Piano-midi.de a v MuseData se nachází většina tónů, které lze zahrát na klavír (viz obrázek 3.3). Délka jednotlivých skladeb v datasetech je kromě Nottinghamu rozdílná (viz obrázek 3.4).

Datové soubory[1] používané v této práci jsou převzaty z článku *Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription* [2]. Soubory jsou k dispozici ve formě MIDI a nebo ve formě piano-rollu, který je použit pro tuto práci. Jednotlivé datové soubory jsou dále rozděleny do tří množin (trénovací, validační, testovací).

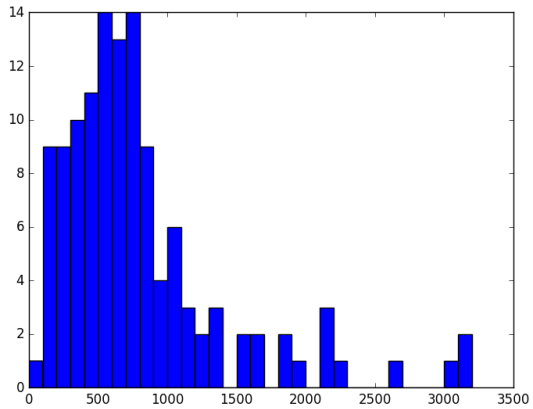
Tabulka 3.1: Počet jednotlivých vzorků v datasetech rozdělených na trénovací, validační a testovací vzorky.

Dataset	Trénovací	Validační	Testovací
Piano-midi.de	76k	8.5k	19k
MuseData	245k	83k	64k
JSBChorales	14k	4.6k	4.7k
Nottingham	177k	45k	44k

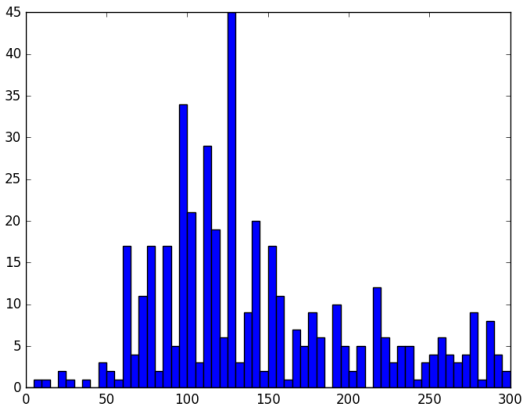
3.3 Tvorba trénovacích vzorů pro neuronovou síť

Při trénování dopředných sítí je síti vždycky předložen 1 vzorek jako vstup a 1 vzorek, který za ním následuje jako odpovídající výstup.. Naproti tomu při trénování rekurentních neuronových sítí se síti předkládá 1 vzorek s pamětí 16 předchozích vzorků. Při následném generování sítí jsou síti vždy postupně předkládány vzorky z původní skladby.

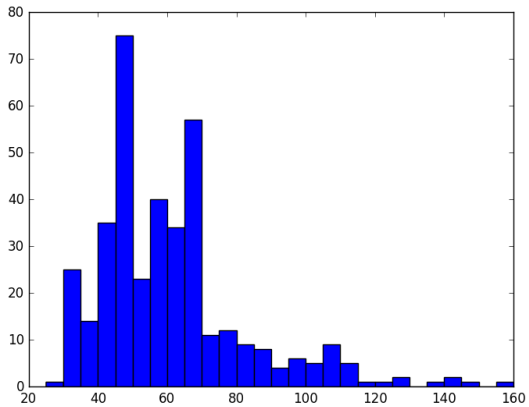
[1] www-etud.iro.umontreal.ca/boulanni/icml2012



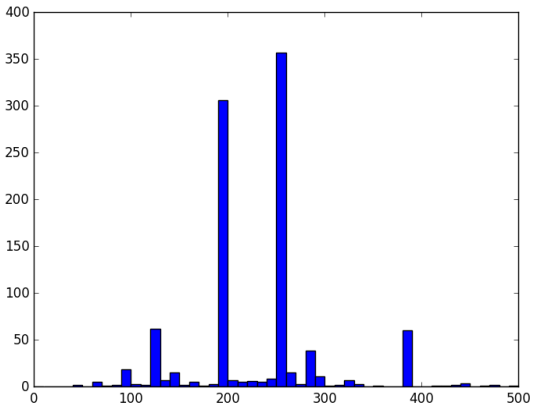
(a) Piano-midi.de



(b) MuseData

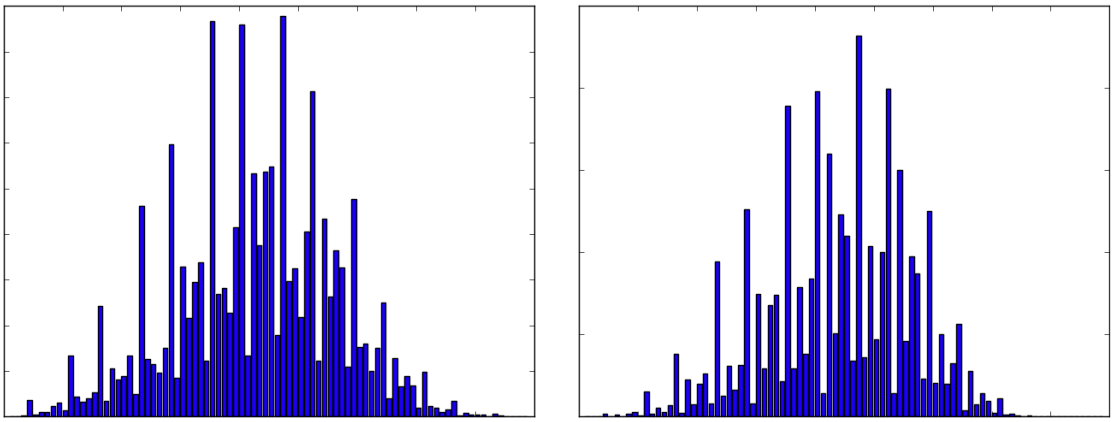


(c) JSBChorales



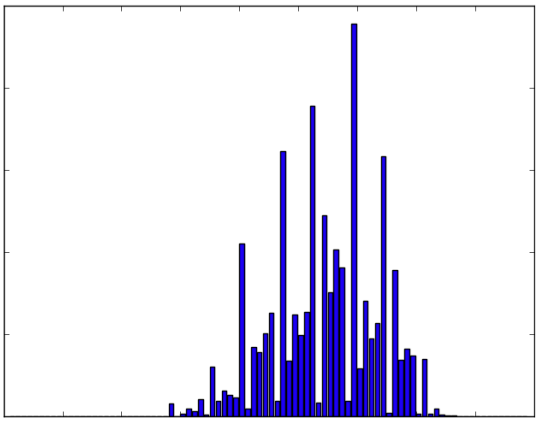
(d) Nottingham

Obrázek 3.3: Rozložení délky skladeb v jednotlivých datasetech: Lze si povšimnout, že kromě Nottinghamu, který obsahuje jen velmi jednoduché a krátké lidové skladby, mají skladby v datasetech velmi rozdílné délky.

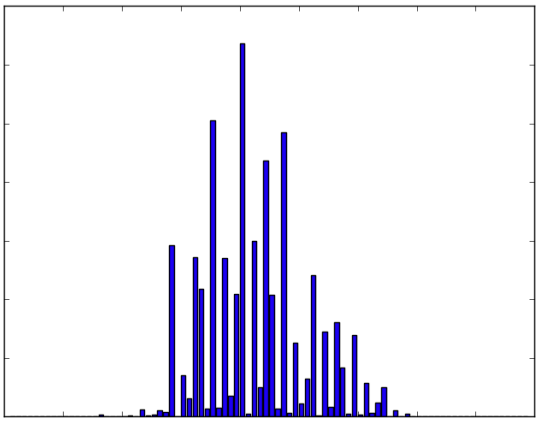


(a) Piano-midi.de

(b) MuseData



(c) JSBChorales



(d) Nottingham

Obrázek 3.4: Četnost použitých tónů v jednotlivých datasetech. Velmi vysoké tóny a velmi nízké tóny nejsou příliš používané. Také lze pozorovat mezery mezi nejčastější používanými tóny, tyto mezery vznikají použitím různých tónin.

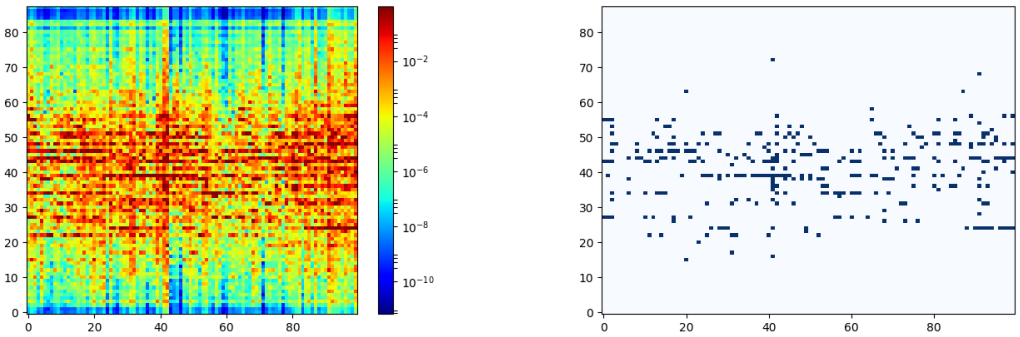
Kapitola 4

Experimenty s konfigurací neuronové sítě

Při vytváření neuronové sítě a jejím následujícím učení je potřeba vzít v úvahu spoustu hyper-parametrů a vhodným upravením těchto parametrů lze zvýšit schopnosti neuronové sítě při modelování sekvencí.

4.1 Proces samplování výstupu

Výstupem neuronové sítě je vektor obsahující reálná čísla z množiny $\langle 0, 1 \rangle$; každou klapku ale lze buď zahrát úplně, nebo nezahrát vůbec. Proto je potřeba převést vektor výstupu neuronové sítě na vektor, který bude možné následně převést do MIDI formátu. K tomu je potřeba vygenerovat náhodný vektor čísel z množiny $\langle 0, 1 \rangle$, celý vektor je potom porovnán, pokud bylo vygenerováno číslo, které je nižší než odpovídající hodnota výstupního vektoru, je vektoru nastavená na danou pozici 1.



(a) Výstup neuronové sítě

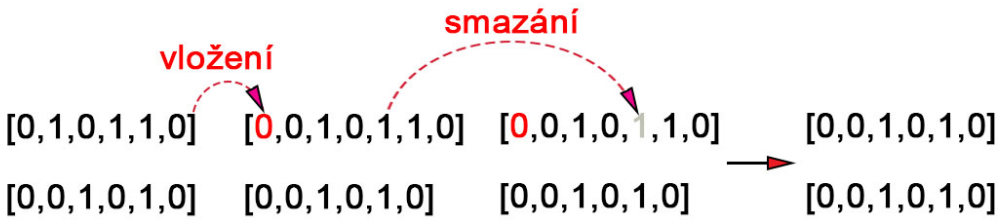
(b) Výsledný vzorek

Obrázek 4.1: Barevný graf (a) značí výstup neuronové sítě a pravděpodobnost, že se daný tón má hrát, (b) značí modré tečky na místě, kde se hraje daný tón. Na obrázku si lze všimnout, že kvůli náhodě můžeme po vzorkování získat pár zahraných klapek, které vůbec neodpovídají zbytku skladby.

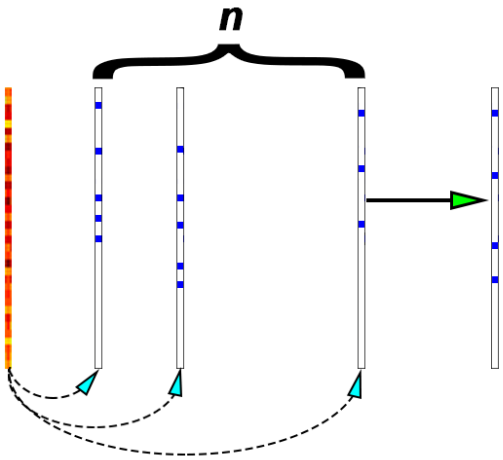
4.2 Metriky výkonu neuronové sítě

K ohodnocení výkonu neuronové sítě jsem použil dvě hlavní metriky: chybu a přesnost. Funkce chyby (viz rovnice 2.3) se vypočítá ještě před vzorkováním a porovnává samotný výstup neuronové sítě s očekávaným výstupem.

Přesnost se počítá pomocí *Levenshteinovy vzdálenosti* a jedná se o přesnost vzorkovaných not. Levenshteinova vzdálenost je vzdálenost dvou řetězců, definovaná jako minimální počet operací vkládání, mazání a substituce takových, aby po jejich provedení byly zadané řetězce totožné. Byla navrhnuta v roce 1965 [7]. Protože je výstup neuronové sítě vzorkování a jedná se o náhodný proces, je potřeba vytvořit více vzorků a teprve z nich určit průměrnou přesnost (viz. obrázek 4.3).



Obrázek 4.2: Příklad Levenshteinovy vzdálenosti. Každá operace vkládání, mazání a substituce má váhu 1, sečtením všech těchto operací získáme Levenshteinovu vzdálenost, která je v tomhle uvedeném příkladu 2



Obrázek 4.3: Každý jeden úsek výstupu je vzorkován náhodným vektorem (světle modrá šipka), je vytvořeno n vzorků, u kterých je vypočítána Levenshteinova vzdálenost vůči původnímu vzorku ze skladby, průměrem těchto vzorků je pak vypočítána Levenshteinova vzdálenost daného výstupu.

Pomocí vypočítáné Levenshteinovy vzdálenosti lze vypočítat accuracy:

$$\text{Přesnost} = 1 - \frac{\text{Levenshteinova vzdálenost}}{\text{počet aktivních klapek v původním výstupu}} \tag{4.1}$$

V případě, že síť vygenerovala jakkýkoliv výstup, ale v původním výstupu bylo pouze ticho (0 aktivních klapek), je jevu přiřazena přesnost = 0, aby nedocházelo k dělení nulou.

V tabulce 4.1 jsou hodnoty validační přesnosti a chyby z práce [2]

Tabulka 4.1: Výsledky jiné práce s neuronovými sítěmi na stejných datasetech[2], při použití RNN vrstev

Dataset	Validační přesnost	Validační chyba
Piano-midi.de	19.33 %	8.37
MuseData	23.25 %	8.13
JSBChorales	28.46 %	8.71
Nottingham	62.93 %	4.46

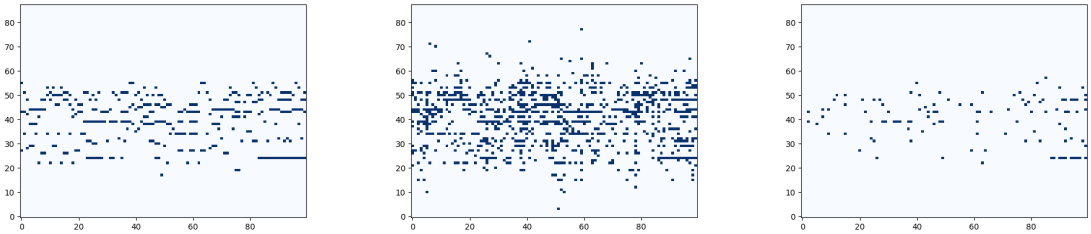
4.3 Skryté vrstvy v modelu

Protože by výpočetní schopnosti sítě skládající se pouze ze vstupní a výstupní vrstvy nebyly dostatečné, je potřeba přidat skryté vrstvy. Začal jsem přidáním jedné dopředné vrstvy (viz tabulku 4.2)

Tabulka 4.2: Testování dopředných neuronových sítí na datasetu Piano-midi.de. Model se skládá ze vstupní a výstupní vrsty o 88 neuronech a jedné skryté dopředné vrstvy o proměnném počtu neuronů.

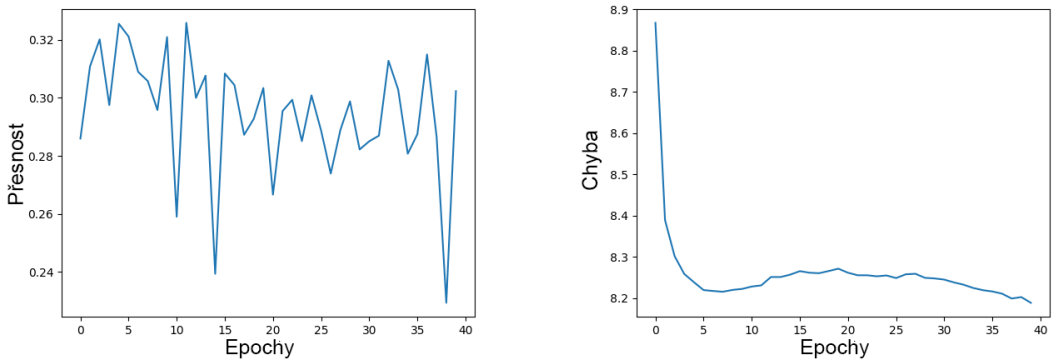
Počet neuronů	Přesnost	Validační přesnost	Validační chyba
256	32.590 %	27.421 %	9.454
512	32.764 %	27.612 %	9.335
1024	29.218 %	23.737 %	9.442
8224	25.662 %	22.054 %	11.236

I když objektivní metriky působí dobře (viz tabulku 4.2), po poslechnutí hudby a zhodnocení výstupů se jednalo pouze o náhodné hodnoty, které se ještě s každou epochou prudce měnily (viz obrázek 4.4). Bez znalosti předchozích tónů měla neuronová síť problémy generovat další tóny a všem tónům přikládala buď příliš velkou váhu, nebo příliš malou váhu.



(a) Původní skladba (b) Vzorek po 39 epochách (c) Vzorek p po 40 epochách

Obrázek 4.4: Dopředné neuronové sítě měly problémy s učením a jejich vzorkovaný výstup se s každou epochou prudce měnil. Problémem může být špatný krok učení. Stejně jako u všech ostatních neuronových sítí, byl jako optimalizační algoritmus použit RMSprop. Podle mého názoru tedy malá dopředná síť vůbec nezvládá učení hudebních dat. Ilustrační obrázek je FFNN se skrytou vrstvou s 256 neurony při vstupu Piano-midi.de, která měla nejlepší accuracy z dopředných vrstev (viz. tabulka 4.2).



(a) Trénovací přesnost (b) Trénovací chyba

Obrázek 4.5: Pro neuronové sítě je typické chování zvyšování přesnosti a snižování loss při porovnávání na trénovacích datech. U dopředných sítí ale nedocházelo ke stoupání přesnosti. Zároveň lze sledovat nezvyklé chování funkce chyby, která se začala na 10 epoše zvyšovat a postupně se od 20 epochy začala snižovat.

V hudbě nejde pouze o to, jak zní jeden samotný tón, ale o jeho zasazení do celé skladby. Proto by rekurentní neuronové sítě měly poskytovat mnohem lepší výsledky, než ty dopředné.

Důležitým rozhodnutím je vybrat správný počet neuronů pro LSTM vrstvy. Po neúspěchu s učením dopředných sítí, jsem u LSTM vzal jako hlavní metriku úspěchu přesnost na testovacích datech. Validační přesnost by měla jít vylepšit přidáním dropout vrstev (viz kapitola 4.5)

Z tabulky 4.3 lze vypozorovat, že 1024 neuronů mělo nejmenší validační chybu, bohužel vrstvy s velkým počtem neuronů byly nestabilní a během trénování často docházelo k pádům sítě, která se nedokázala většinou učit po déle než 12 epoch a menší sítě ji po delší době učení překonali.

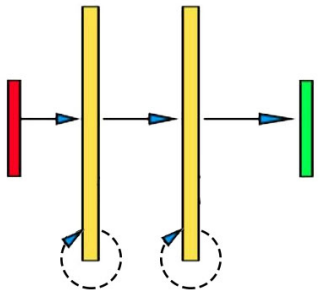
Tabulka 4.3: Vliv počtu neuronů ve vrstvě na přesnost a chybu, 2 LSTM vrstvy, trénováno na Piano-midi.de

Počet neuronů	Přesnost	Validační přesnost	Validační chyba
256	63.256 %	24.960 %	13.675
512	70.346 %	25.411 %	12.163
1024	62.374 %	26.218 %	11.431

Nejlepší validační přesnost a validační chyba byla u 3 vrstev, o hodně lepší přesnost se skoro stejnou validační přesností a validační chybou byla u 2 vrstev (viz. tabulka 4.4).

Tabulka 4.4: Vliv počtu skrytých vrstev na přesnost a chybu, 512 neuronů ve vrstvě trénované na Piano-midi.de

Počet vrstev	Přesnost	Validační přesnost	Validační chyba
1	66.241 %	21.821 %	14.545
2	70.346 %	25.611 %	12.163
3	62.374 %	26.218 %	11.831

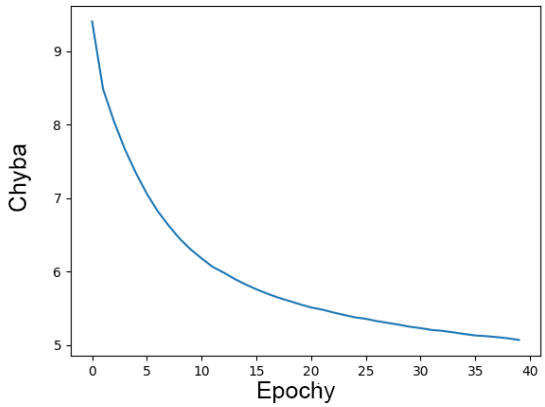


Obrázek 4.6: Používaný model neuronové sítě, (červená) vrstva značí vstupní vrstvu o šířce odpovídající vstupům (88), většina mých modelů se skládala ze dvou (žlutých) LSTM vrstev o proměnné šířce, výstup neuronové vrstvy je potřeba znovu převést na "hudební formát"s (zelenou) výstupní vrstvou.

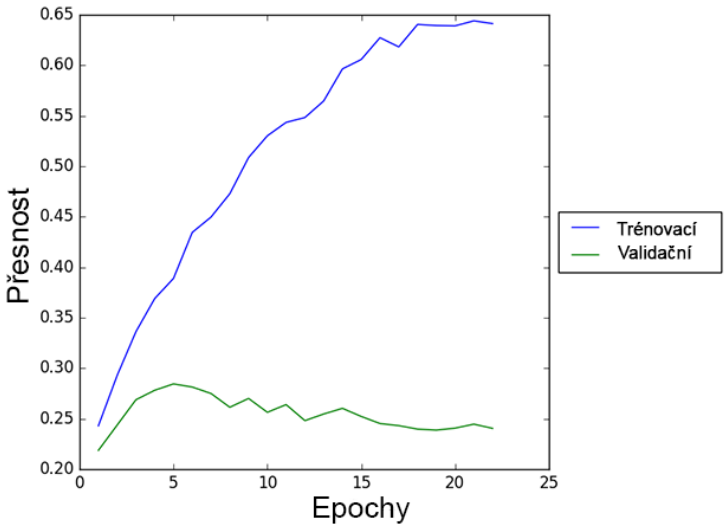
4.4 Počet epoch a přetrénování

Jeden průchod neuronové sítě všemy trénovacími daty se nazývá epocha. Protože se síť nedokáže přizpůsobit sekvencím v datech tak rychle, je potřeba provést těchto epoch více. Přidáváním epoch se zvyšuje schopnost sítě modelovat sekvence, po určitém počtu epoch se ale validační přesnost sítě může zhoršovat.

Zároveň lze vypozorovat, že při většině pokusů *chyba* stále klesala a *přesnost* stále stoupala, zatímco *validační přesnost* nekonvergovala (viz. obrázek 4.8). Přeučení je tedy reálným problémem neuronových sítí a s ním přichází i problém odhadovat kvalitu neuronové sítě. V případě, že se síť velmi přeučila může vytvořit výstup, který se po vzorkování bude zdát velmi příjemný na poslech, v případě přeučené sítě, se ale může jednat pouze o kopii původní skladby a to není požadované chování sítě.

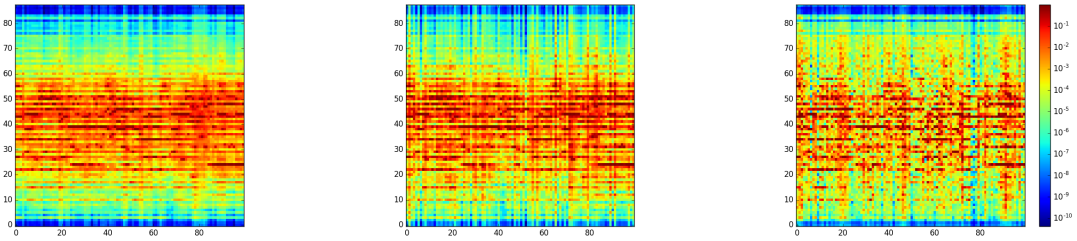


Obrázek 4.7: Změna hodnoty *chyby* po prvních 40 epochách dvou LSTM vrstev o šířce 512 při trénování na dataset Piano-midi.de, první epochy měly největší vliv na neuronovou síť a s každou další přibývající epochou se tento vliv zmenšoval.



Obrázek 4.8: Ukázka vývoje trénovací přesnosti a validační přesnosti neuronové sítě s jednou 512 LSTM, trénováno na Piano-midi.de. Z grafu lze vidět neustále zlepšení neuronové sítě na testovacích datech, zatímco schopnosti neuronové sítě na validačních datech se s přibývajícími epochami silně zhoršují.

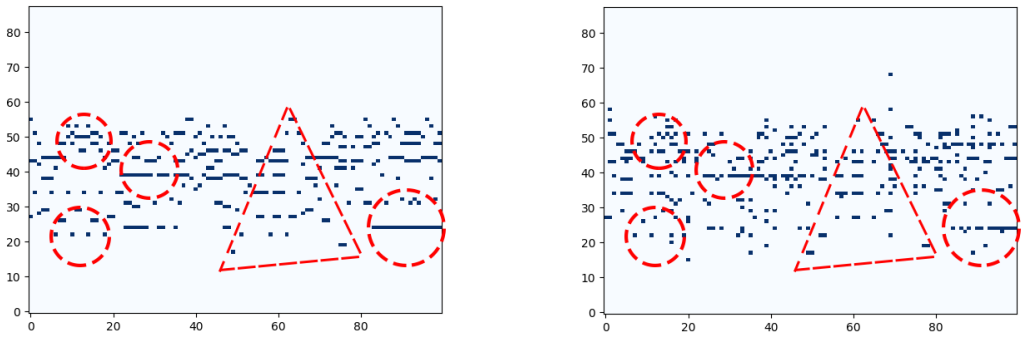
Zároveň lze také sledovat pokles validační přesnosti a validační chyby (viz tabulka 4.5), protože síť se přílíš přizpůsobila trénovacím datům, zlepšovala svoje schopnosti generování na nich, ale zhoršovala schopnosti generování dat, které budou jen trošku odlišné. Jevu, kdy se síť příliš naučí trénovací data, se říká *overfitting* a jeho zmírnění je popsáno v další sekci.



jedna epocha (a) pět epoch (b) čtyřicet epoch (c)

Obrázek 4.9: Výstup neuronové sítě při stejném vstupu z validačních dat, červené barvy značí velkou šanci na výskyt tónu, modré barvy naopak nižší. Po první epoše (a) získala síť schopnost rozeznat, které tóny jsou v daném datasetu hrány, ale až po delším trénovacím čase (c) se síť naučila přidávat důležitost daným tónům v závislosti na předchozích hraných tónech.

Vygenerovaný výstup se také značně zlepšuje s přibývajícími epochami (viz obrázek 4.9). Na vzorkovaném výstupu z neuronové sítě můžeme porovnat vzorkovaný výstup s požadovaným výstupem (viz obrázek 4.10).



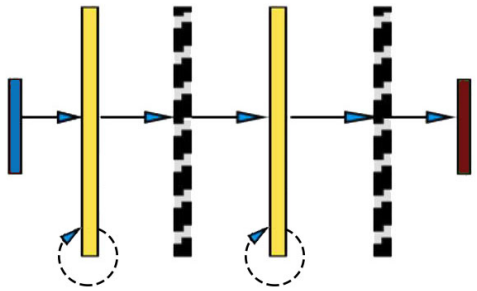
(a) Původní skladba (b) Vzorkovaný výstup NN

Obrázek 4.10: V obrázcích (a) a (b) lze najít určité spojitosti. Zároveň to působí, že síť zbytečně nevybírá tóny, které se vůbec nehodí hrát, a jen pár se jich dostalo do výstupu, kvůli procesu vzorkování. Příklad na datasetu Piano-midi.de. Vzorkovaný výstup je mnohem lepší než u dopředných neuronových sítí (viz. obrázek 4.4)

4.5 Dropout

K overfittingu (přeučení) dochází, pokud jsou síti předkládána pořád stejná data a síť se příliš naučí rozpoznávat pouze vstupní data a začne mít problémy s modelováním jiných sekvencí. Jako jedno z řešení overfittingu byl navrhnut dropout [10].

V mé práci je dropout využíván k tomu, aby zamezil vysoké kooperaci jednotlivých neuronů, vypínáním náhodných neuronů z neuronové sítě. Každý neuron je vypnutý (včetně jeho spojení) z neuronové sítě s určitou pravděpodobností p. Díky náhodnému vypínání neuronů se zabraňuje, aby se síť příliš přizpůsobila vstupním datům.



Obrázek 4.11: Model neuronové sítě po přidání dropout vrstev, bílo-černé bloky představují dropout, zabraňují průchodu signálu z náhodných neuronů předešlé LSTM vrstvy.

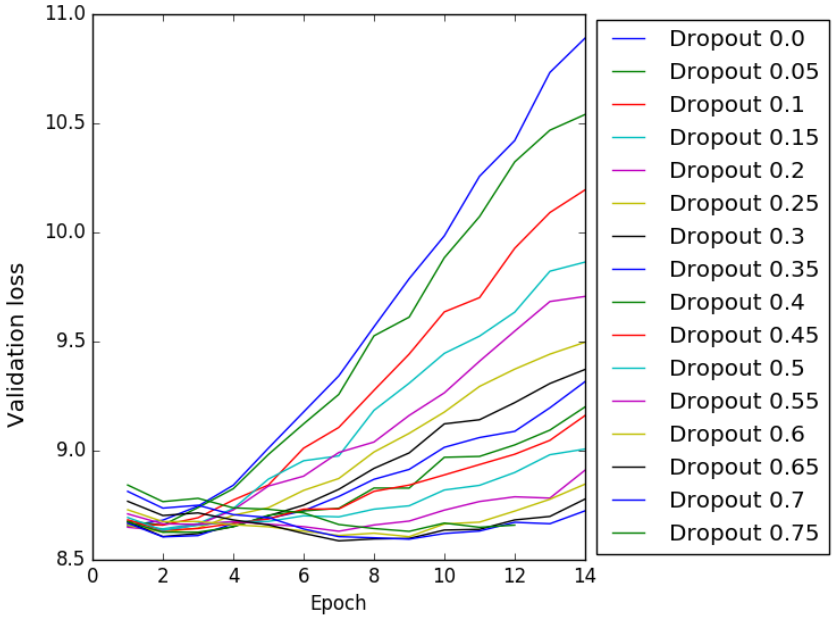
Tabulka 4.5: Vliv velikosti dropoutu, modely vybírány podle největší dosažené validační přesnost, dvě 512 LSTM vrstvy trénovány na Piano-midi.de

Dropout	Trénovací přesnost	Validační přesnost	Validační chyba
0.0	50.192 %	26.014 %	10.143
0.1	48.891 %	28.444 %	9.366
0.2	45.557 %	27.369 %	9.345
0.3	29.750 %	22.205 %	8.809
0.4	21.345 %	21.239 %	8.635

Přidáním dropoutu do modelu byla velmi lehce zvýšena přesnost na validačních datech. I přes to, pořád docházelo k přeučení. Dropout 0.1 se prokázal být nejvhodnější pro danou úlohu. Dalším zvyšováním hodnoty dropout se výpočetní schopnosti sítě zhoršovaly.

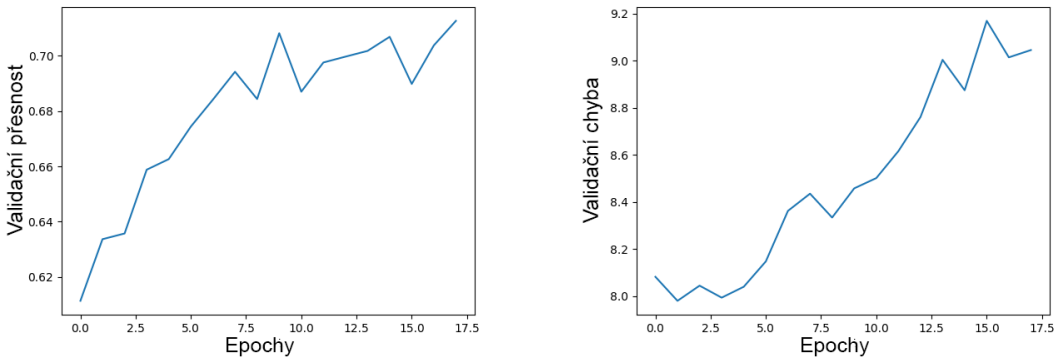
Tabulka 4.6: Zhodnocení modelu se dvěmi vrstvami o šířce 512 neuronů a dropoutem 0.1 na všech datasetech; vybrán model podle nejlepší validační přesnost, při všech trénování se neuronová síť přeučovala a funkce přesnost tedy stále rostla.

Dataset	Přesnost	Validační přesnost	Validační loss
Piano-midi.de	38.891 %	28.444 %	9.088
MuseData	41.372 %	30.714 %	10.118
JSBChorales	31.193 %	31.193 %	14.050
Nottingham	86.957 %	71.264 %	9.044



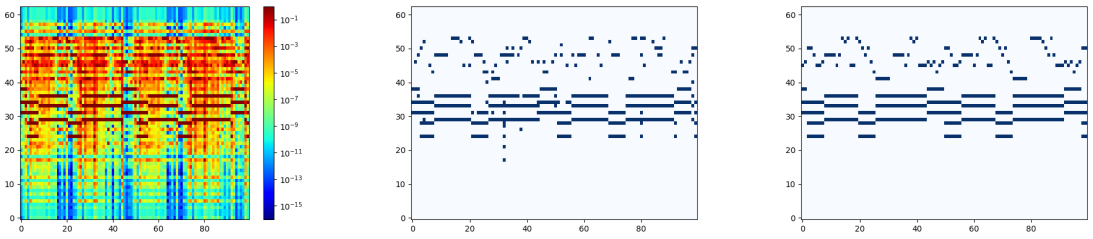
Obrázek 4.12: Různé hodnoty dropoutu pro neuronovou síť (dvě LSTM vrstvy s 256 neurony) trénovanou na datech Piano-midi.de. I přes svou neúčinnost ve zvyšování validační přesnosti se dropout ukázal být velmi užitečný pro snižování validační chyby.

Při porovnání dat z tabulek 4.6 a 4.1 lze zjistit, že moje validační přesnost je u všech případů vyšší, zároveň je vyšší i validační loss. Toto chování je poněkud podivné, protože oba používáme stejnou přesnost vypočítanou pomocí Levenshtainovy vzdálenosti. V případě [2] byla používaná jednoduchá RNN, zatímco v mé práci je použito LSTM. Například u validační chyby Nottinghamu, který se v obou testování projevil nejlépe, moje validační přesnost stoupá zároveň s validační chybou (viz obrázek 4.13). V datasetu Nottingham také dokázala neuronová síť modelovat hudební sekvence odpovídající původní skladbě (viz. obrázek 4.14).



(a) Validační přesnost (b) Validační chyba

Obrázek 4.13: Porovnání validační přesnosti a validační chyby neuronové sítě s dvěmi 512 LSTM a dropoutem 0.1 na datasetu Nottingham. S přibývajícími epochami obě hodnoty lineárně rostou, i když by se validační loss měla alespoň po určitou dobu snižovat.



(a) Výstup neuronové sítě (b) Navzorkovaný výstup (c) Původní část skladby

Obrázek 4.14: Srovnání výstupu neuronové sítě (a), navzorkování tohoto výstupu (b) a původní části skladby (c). V (a) lze pozorovat šance zahrání jednotlivých tónů v čase. Místa, která se nacházejí v originální skladbě (c), neuronová síť se naučila velmi přesně modelovat hudbu v datasetu Nottingham. Kromě pár navzorkovaných tónů navíc se vzorkovaný výstup a původní skladba příliš neliší.

Kapitola 5

Jednoduchá aplikace pro generování hudby

Aplikace slouží k jednoduché prezentaci možností neuronových sítí v oblasti modelování hudebních sekvencí. Aplikace se skládá pouze z velmi základních prvků pro úpravu hudby.

5.1 Návrh aplikace

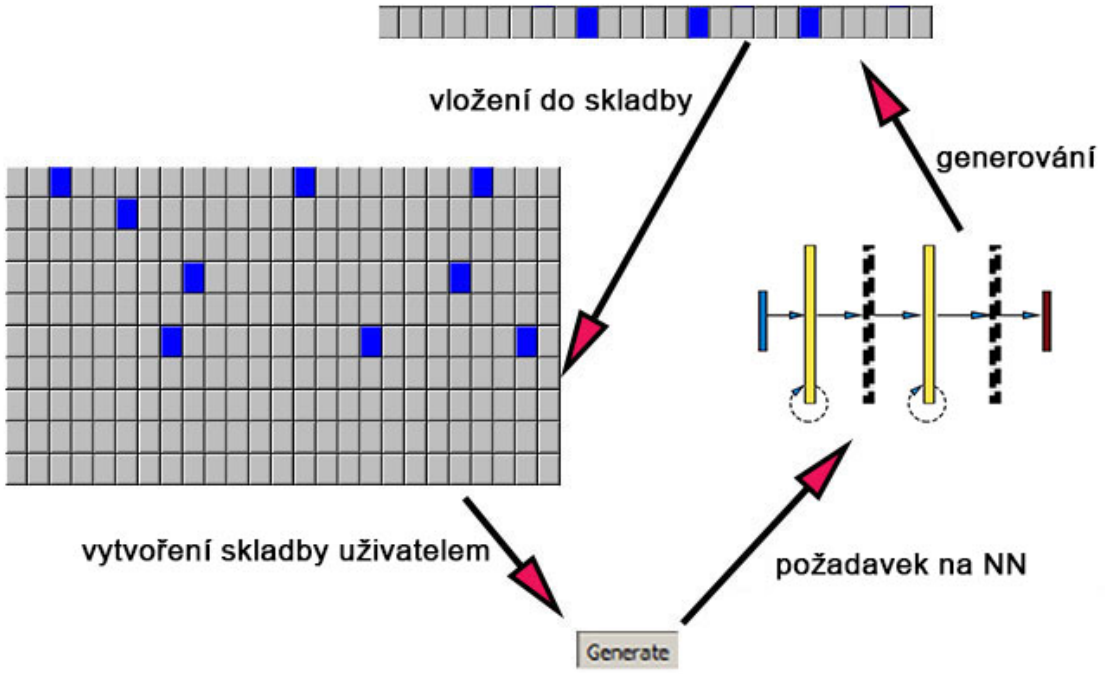
Převážná část aplikace se skládá z reprezentace 88 klapek piána a jednoduché možnosti přehrát vlastní hudbu a zapnutí neuronové sítě pro generování další hudby. Skladba je uložena v Python seznamu a může být hned předložena neuronové síti.

5.2 Externí závislosti

Aplikace v Pythonu; i přes fungování aplikace na více platformách je doporučováno používat Windows, protože na Linuxu může přehrávání hudby v aplikaci dělat problémy. Neuronová síť pro moji aplikaci byla vytvořena a natrénována pomocí knihoven `Keras` a `TensorFlow`. Funkce těchto knihoven jsou využívány při samotném generování hudby pro uživatele, je tedy potřeba mít ve svém python prostředí nainstalovány obě tyto knihovny. Knihovny jsou distribuovány ve spoustě verzí a z důvodu změn funkce pro načítání modelu, je pro správný běh doporučovaný `TensorFlow` 1.0.1 a `Keras` 2.0.3. Uživatelské rozhraní aplikace bylo vytvořeno pomocí knihovny `tKinter`. Generování vysledného midi souboru zajišťuje knihovna `miditime`, následné přehrání hudby je zajištěno pomocí knihovny `pygame`.

5.3 Neuronová síť v aplikaci

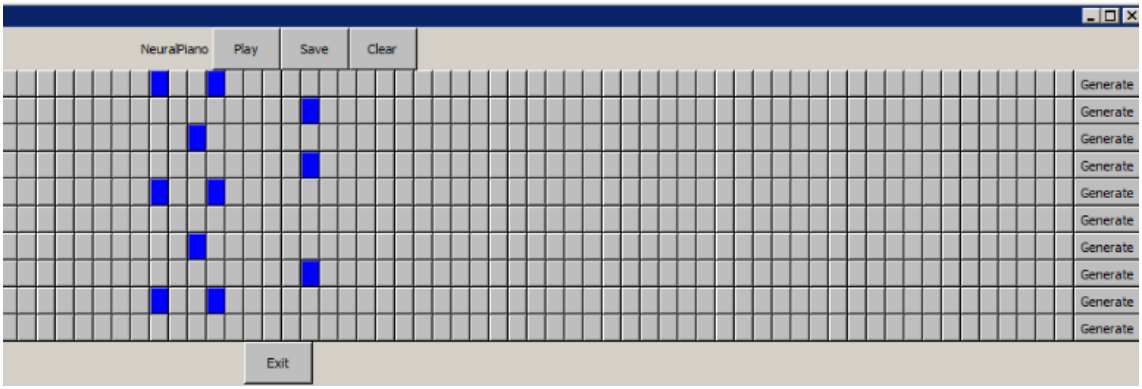
Pro aplikaci byla zvolena neuronová síť s dvěma LSTM o šířce 512 neuronů a dropoutem 0.1 za každou z těchto vrstev. Tato neuronová síť byla naučena na datasetu Nottingham, který poskytoval nejlepší přesnost na validačních datech. Aplikace ale bude fungovat s jiným modelem, který je uložený ve stejné složce jako hlavní skript. Jenom je potřeba, aby používaný model byl natrénován na stejné verzi Kerasu 2.0.3.



Obrázek 5.1: Návrh fungování aplikace. Uživatel vytvoří skladbu v UI připomínající piáno, skladba je uložena v Python seznamu. Po ztlačení tlačítka *generate* je seznam poslán neuronové síti jako vstup, její navzorkovaný výstup je pak přidán do skladby.

5.4 Uživatelské rozhraní

GUI ukazuje zjednodušené piano v čase. Osa x značí konrétní hrající klapky klavíru v jednom čase. Osa y potom značí čas, začínající z vrchní části dolů. Pomocí tlačítek *generate*, lze zadaný řádek nechat vygenerovat neuronovou sítí. Tlačítko *play* přehraje skladbu v aplikaci.



Obrázek 5.2: Uživatelské prostřední aplikace NeuralPiano

5.5 Možné vylepšení aplikace

Jak již bylo zmíněno v předchozí části této práce, aplikace je opravdu velmi jednoduchá a neumožňuje editaci hudby srovnatelnou s jinými komerčními aplikacemi. Jako rozšíření aplikace by bylo možné vylepšit uživatelské rozhrání aplikace, které momentálně slouží pouze k předvedení schopností rekurentních neuronových sítí. Zároveň uživatel nemůže načíst svoji hudbu, která byla vytvořena v jiném editačním softwaru a musí svoji hudbu znovu vytvořit v této aplikaci.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo natrénovat rekurentní neuronovou síť na klasickou klavírní hudbu. Každý dataset obsahoval alespoň 7 hodin hudby, dohromady datasety obsahovali přibližně 67 hodin hudby. Dopředné sítě měli problémy přizpůsobit se hudbě a jeden vzorek pro modelování hudby nebyl dostatečný.

LSTM sítě byly velmi schopné přizpůsobit se trénovacím datům v porovnání s původním článkem [2], ze kterého byly vypůjčeny datasety pro trénování sítě. Na datasetu Nottingham, který se skládá z lidové hudby síť dokázala získat až 71 % validační přesnost. Všechny validační přesnosti byly v průměru o 6.91 % vyšší než v původním článku.

Jako ideální konfigurace se ukázala síť s dvěmi skrytými LSTM vrstmi o šířce 512 neuronů a dropoutem 0.1. Napsal jsem článek o generování hudby na Excel [8]

Jako ukázku možnosti neuronových sítí jsem vytvořil jednoduchý program, ve kterém může uživatel poslat své noty síti a ta přidá do skladby další notu. Program je dostupný jako skript v Pythonu.

Při současném přístupu je možné, že navzorkovaný výstup neuronové sítě bude znít špatně, protože po navzorkování získáme tón, který se k ostatním nehodí. Tento problém by se dal vyřešit úpravou samotných vzorků, aby se více podobali notám hraným v datasetech.

Dalším možným rozšířením této práce by mohla být aplikace, která by se naučila celé skladby a dokázala podle nich určit; o jaký styl hudby se jedná, případně najít podobné melodie pro uživatele. V tomto případě by to bylo složitější a musel by se změnit způsob zpracovávání dat, aby bylo možné pracovat s více nástroji než jen s pianem.

Literatura

[1] LSTM Networks for Sentiment Analysis. http://deeplearning.net/tutorial/lstm.html, online; navštíveno 10.05.2017.

[2] Boulanger-Lewandowski, N.; Bengio, Y.; Vincent, P.: Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, editace J. Langford; J. Pineau, New York, NY, USA: ACM, 2012, s. 1159–1166.

[3] Elman, J. L.: Distributed Representations, Simple Recurrent Networks, And Grammatical Structure. *Machine Learning*, ročník 7, č. 2, 1991: s. 195–225, ISSN 1573-0565, doi:10.1023/A:1022699029236.

[4] Hinton, G.: Overview of mini-batch gradient descent. Online; navštíveno 11.05.2017. URL http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[5] Hochreiter, S.; Schmidhuber, J.: Long Short-Term Memory. *Neural Comput.*, ročník 9, č. 8, Listopad 1997: s. 1735–1780, ISSN 0899-7667, doi:10.1162/neco.1997.9.8.1735.

[6] Jordan, M. I.: Serial Order: A Parallel, Distributed Processing Approach. In *Advances in Connectionist Theory: Speech*, editace J. L. Elman; D. E. Rumelhart, Hillsdale, NJ: Erlbaum, 1989.

[7] Levenshtein, V.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, ročník 10, 1966: str. 707.

[8] Majer, M.: Applying Recurrent Neural Network To Music Generation. 2017. URL http://excel.fit.vutbr.cz/submissions/2017/061/61.pdf

[9] McCulloch, W. S.; Pitts, W.: Neurocomputing: Foundations of Research. kapitola A Logical Calculus of the Ideas Immanent in Nervous Activity, Cambridge, MA, USA: MIT Press, 1988, ISBN 0-262-01097-6, s. 15–27.

[10] Srivastava, N.; Hinton, G.; Krizhevsky, A.; aj.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, ročník 15, č. 1, Leden 2014: s. 1929–1958, ISSN 1532-4435.

[11] Werbos, P. J.: *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. Wiley-Interscience, 1994, ISBN ISBN: 978-0-471-59897-8.