UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Music Generation Powered by Artificial Intelligence

Author:
Oleh SHYSHKIN

Supervisor:
Dr. Juan Pablo MALDONADO LOPEZ

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2019

Declaration of Authorship

I, Oleh SHYSHKIN, declare that this thesis titled, "Music Generation Powered by Artificial Intelligence" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Abstract

Faculty of Applied Sciences

Master of Science

Music Generation Powered by Artificial Intelligence

by Oleh SHYSHKIN

Music is an essential part of human life in our days. Despite a long history of the phenomena people still explore it and expand the new horizons. For the last ten years quality of computer-generated music significantly improved. State of the art machine learning models like PerformanceRNN can perform music close to a human level. However, it is hard to deal with a generation of long-term music for the systems. In work, we apply a TCN model to a generation music task and evaluate the quality of the music. We show that the models have a significantly better performance than a baseline model for a long-term music generation task. However, it has own weak points in musicality and time generation. We also discuss possible options to resolve the issues.

Acknowledgements

The work wouldn't be possible without people who believe in me and support me all the time. Many thanks to Ukrainian Catholic University for the unique and inspirational environment. Special thanks to my supervisor Pablo Maldonado who led and supported me during the journey.

Also, I would like to thank Magenta team for such a beautiful project which prepared a good foundation and infrastructure for exploring human creativity and music. Especially, I am grateful to Ian Simon and Colin Raffel for sharing their knowledge and providing useful feedback.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	**C**onvolutional **N**eural **N**etwork
MIDI	**M**usical **I**nstrument **D**igital **I**nterface
NADE	**N**eural **A**utoregressive **D**istribution **E**stimation
LSTM	**L**ong **S**hort-**T**erm **M**emory
TCN	**T**emporal **C**onvolutional **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
RL	**R**einforcement **L**earning

Chapter 1

Introduction

Music is an important part of human life. It surrounds us everywhere: movies, advertisements, games, shops, we listen to it when we study or when we work. It has been developing by people for thousands of years. And as one of the manifestations of human creativity, music is an interesting subject for study.

From a scientific perspective, solving the task has to help us better understand a human mind and intelligence. From an application side, a better music generation can be used in multiples aspects of human activities like writing movies/games soundtracks, creating interactive tools for learning music theory, music psychotherapy, etc.

The programs that generate music exists for a long time.Band-in-a-Box[1] is a system which generate jazz, blues and rock instrumental solos with almost no human interaction. Another example is Impro-Visor, which uses a stochastic context-free grammar to generate phrases and complete solos[2].

However, for the last few centuries, a number of music genres dramatically increased. "Musicmap" [3] project claims that today there are nearly 234 genres of music, but some sources count even bigger numbers of genres. The number depends on classification methodology and level of abstraction, but it does not give enough generalization power. Often times a genre has its own set of rules and in many cases, the set is neither strict or unique across all genres. It makes a uniforming of heuristic rules-based approaches for music generation a pretty challengeable task. On the other hand, the amount of data and computational power of our days' computers allows us to use modern machine learning tools. Inspired by PerformanceRNN (Oore et al., 2018) and WaveNet (Oord et al., 2016) recent results we want to understand how good is a convolutional neural network in a task where sequential neural networks have a dominant position.

WaveNet has already shown that it can deal with enormous long sequences, and PerformanceRNN has demonstrated that neural networks can generate human kind music. However, PerformanceRNN is an example of usage of a classical sequential neural networks architecture.

The Machine Learning Department of Carnegie Mellon University has proposed generic convolutional and recurrent architectures for sequence modeling named temporal convolutional network (Bai, Kolter, and Koltun, 2018). The architecture is an attempt to generalize to a single unit the best practices which are used in the best convolutional architectures like WaveNet. Also, according to the article, it outperforms all classical sequential models in music generation task. On the other hand, TCN generated music is not compared to another generated music by human what is crucial because a human is the main consumer.

[1] **https://www.pgmusic.com/**.
[2] **https://www.cs.hmc.edu/~keller/jazz/improvisor/**.
[3] https://musicmap.info/

The work is an exploration of TCN based models applying to a music performance generation task with respect to the PerformanceRNN model. And it includes the next chapters.

Chapter 1. This is an introduction with a broad overview of music field. We briefly discuss music complexity, existed projects, datasets and motivation for the project.

In **Chapter 2** we review state of the art machine learning models applied to a music generation task as it can give ideas for future researches and experiments. We also point on solid surveys to give a view of what was done in the field.

In **Chapter 3** we describe target models and motivation of chosen architectures.

Chapter 4 describes some technical aspects, experiments details and results. In the section, we compare PerformanceRNN with different models based on TCN.

Chapter 5 is a conclusion. Here we summarize an achieved results and points for improvement.

1.1 Problem statement

Music is a multicultural phenomenon which independently appeared in different place around the globe. One of the definitions of music is the next: "This is an art form consisting of sound and silence, expressed through time". People have been developing music theory and did numerous experiments in discovering human sounds perception. However, until recent years it was hard to analyze music at scale or to get breakthrough results in music generation task. In our opinion there are a few reasons for that:

- Music complexity (level of abstractions, number of variations, and nuances of human music perception)
- Lack of software tools for music analyzing in the past (now we have MIDI data representation, projects like LibROSA, Magenta, AcousticBrainz, etc.)
- Lack of well-structured data in the past (now we have many audio data sets like Million Song Dataset, MAESTRO, Lakh etc; all of them are open sourced and free to use)

In the chapter, we briefly consider all of them and describe a vector of the research.

1.2 The complexity of music generation

A good example of music generation framework was proposed in the survey (Herremans, Chuan, and Chew, 2017). The block schema of the system is shown on a Figure 1.1. The proposed structure has two key components (which can be decomposed on subcomponents):

- Composition
  - Melody
  - Harmony
  - Rhythm
  - Timbre
- Note
  - Pitch
  - Duration
  - Onset time
  - Instrumentation

Each of the element can be considered as a topic for research, so we briefly consider only the key elements to get a better understanding of a variety of approaches in music generation task.

A note and a composition are central concepts in music. **The note** is a physical phenomenon when **the composition** is a high-level abstraction created by human. A music creation relies on human feelings and has to call to emotions that the composer experiences or want the other people experience. We build strong associations

between emotions and audio compositions. It allows us to consider a music writing process as a emotions sequence building through the audio composition, and it leads us to human sound perception topic. There is a hypothesis about how our mind differentiates rhythm patterns (London, 2002) or which brain mechanisms are responsible for music recognition (Large and Crawford, 2002). Nevertheless, there is no universal map between sequences of sounds and human emotions, and there is no guarantee that it is universal if it exists. Hence, it is hard to explain to a machine which music we expect when it is relatively simple to explain to people what is a romantic, scary or funny music.

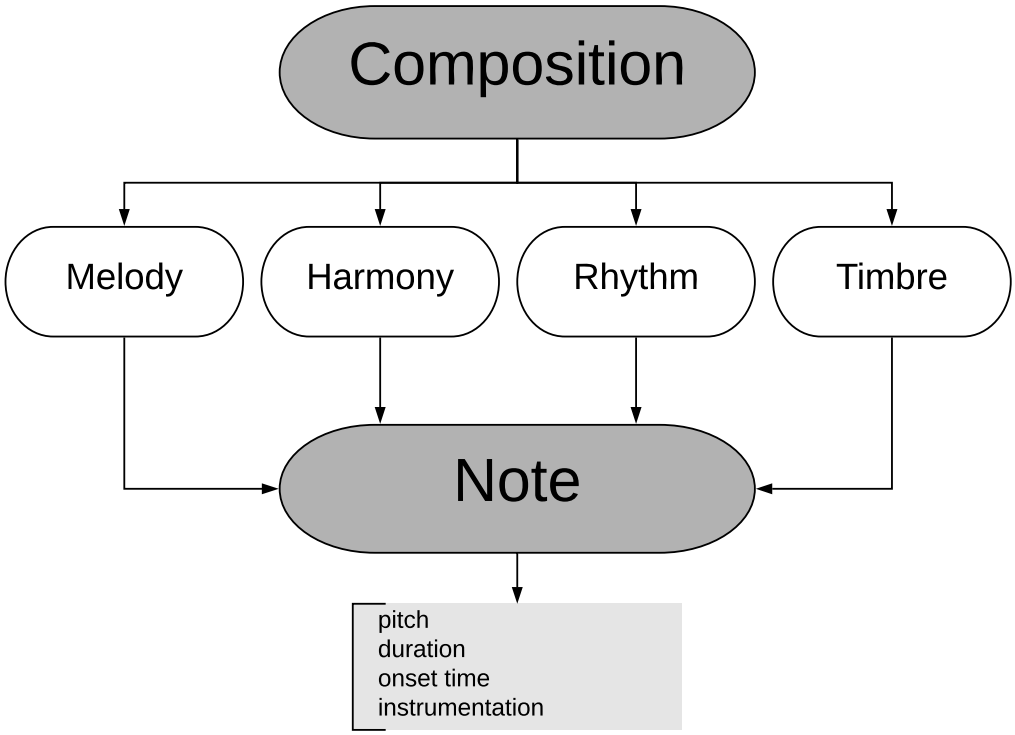

FIGURE 1.1: Music generation framework

One more variant is abandoning sound-emotion connection and try to mimic real music, other words to make it sounds realistic to people. It is a quite popular approach in our day due to the recent success of deep learning in an image generation task and related fields. Music generation in the scope still requires a lot of data, but because we don't need marks or labels we can use real audio tracks almost as it is. However, the biggest problem here is model interpretability and control under a model output. There are many discussion around machine learning "black box" solutions (Ribeiro, Singh, and Guestrin, 2016; Lipton, 2016) and how to deal with it. But the problem is rather common than specific, so despite on these cons, the approach looks the most promising.

Magenta team provides a very colorful and simple example of music complexity from a maths perspective. The task is to generate a monophonic piano melody. At any given time, exactly one of the 88 keys can be pressed down or released, or the player may rest. It can be represented as 90 types of events (88 key presses, 1 release, 1 rest). To simplify the modeling process, we ignore tempo, dynamic and quantize time down to 16th notes. Next, two measures (bars) of music in 4/4 time will have

9032 possible sequences. If we extend this to 16 bars, it will be 90256 possible sequences. Now we can go back and make the piece of music more close to a real piece of music: polyphonic, with multiples instruments, .etc. Because of the high data dimensionality, it is difficult to generate a piece of music using heuristic rules and stochastic models. On the other hand, it can be helpful to tune a final result or lead model training process. One of the Magenta project (Jaques et al., 2016) demonstrated that the RL model based on music theory rules can significantly improve model output.

1.3 The current state of tools for music analyzing

The last but not the least part of each experiment is an analysis of the results. For music generation, the most naive but also the most efficient method is a human evaluation. We will later discuss it in the section 4.3, for now, we are going to consider projects which were developed based on human knowledge about music.

One such project is LibROSA [4]. It provides implementations of a variety of common functions used throughout the field of music information retrieval (MIR). It allows automatic music feature extraction like spectral characteristics, mel-frequency cepstral coefficients (MFCCs), rhythm components, .etc. It can be also used for pre- and post- data processing but this requires some music and signal processing knowledge what is out of the scope for the work.

The next project is AcousticBrainz [5], which aims to crowd source acoustic information for all music in the world and to make it available to the public. This information describes the acoustic characteristics of music and includes low-level spectral information and information for genres, moods, keys, scales, .etc. The goal of AcousticBrainz is to provide music technology researchers and open source hackers with a massive database of information about music. The project is interesting as heuristic music quality metric and can be used to evaluate generated music metadata with respect to AcousticBrainz model.

Another research project is Magenta [6]. It is Google Brain open source project exploring the role of machine learning as a tool in the creative process. Also, it is distributed as an open source Python library, which includes utilities for manipulating source data (primarily music and images). The project has a perfect infrastructure not only for art exploring but also for research goals. Existed data pipelines allow you to transform your data to fit different model input formats and then compare results across the models. That is the main reason why the platform was chosen in the work.

1.4 Available music datasets

A variety of data in our days is exceptional. Nevertheless not each of existed datasets has an appropriate quality and format for research.

One of the most familiar music datasets is a Million Song Dataset [7]. It is a freely-available collection of audio features and metadata for a million contemporary popular music tracks. The core of the dataset is the feature analysis and metadata for one million songs, provided by The Echo Nest. The dataset does not include any

[4] https://librosa.github.io/librosa/
[5] https://acousticbrainz.org/
[6] https://acousticbrainz.org/
[7] https://labrosa.ee.columbia.edu/millionsong/

audio, only the derived features. Hence, the dataset is good for descriptive analysis or kind of supervised learning tasks. To make it suitable for music generation task it is necessary to generate tracks itself using the metadata. It is possible to do using such services "7digital" but it requires extra work with data post-processing.

Lakh MIDI dataset [8] is a collection of 176,581 unique MIDI files, 45,129 of which have been matched and aligned to entries in the Million Song Dataset. The source is used for many MIR tasks, included music generation. It contains many tracks of different styles and with different instruments. This is an advantage of the dataset and a big challenge at the same time. But the difficulties are related more to MIDI side than to dimensionality problem described earlier. MIDI is a technical standard that describes a communications protocol, digital interface, and electrical connectors that connect a wide variety of electronic musical instruments, computers, and related audio devices. Hence, it relies on synthetic sounds libraries for many instruments and has much less capability in sound representation. However, for some instruments like the piano, it is good enough and can be compatible.

One of such source of high-quality piano music data is the MAESTRO dataset (Hawthorne et al., 2018). This dataset contains over a week of paired audio and MIDI recordings from nine years of International Piano-e-Competition. The MIDI data includes key strike velocities and sustain pedal positions. What is also important that each track was played by a human expert and allows us to track us the human component in music performance.

1.5 Vector of the research

Inspiring PerformanceRNN results and regarding task complexity, it looks that it is worth to try to improve the PerformanceRNN in a direction of better capturing composition abstractions like long-term motive, melody, and harmony.

In the next section, we will consider related works and dig deeper on a technical side of the state of the art algorithms.

[8] https://colinraffel.com/projects/lmd/

Chapter 2

Related work

There are many works in the field. Many of them formed baselines and common practices for future researches. This fact stimulated people to try numerous approaches for music generation starting from statistical methods and analysis of music metadata, to applying stochastic modeling and cellular automata. The number of works and their variety were perfectly described in Jose David Fernandez "AI Methods in Algorithmic Composition: A Comprehensive Survey" and Dorien Herremans "A Functional Taxonomy of Music Generation Systems" surveys. However, nowadays deep learning models take more and more popularity due to the tremendous success of the models in different complex tasks like image recognition or machine translation.

2.1 Links to historical overviews

According to Fernandez's survey (Fernández and Vico, 2013), one of the earliest works related to computer generated music was published in the mid-1950. The model uses rule systems and Markov chains. Except the method researchers also explored approaches based on:

- Grammars
- Symbolic and knowledge-based systems
- Evolutionary and population-based theory

Also, a quite popular tool for composers was model which used cellular automata and self-similarity ideas. They produce a raw piece of music with unusual patterns that inspire musicians and help them create new music.

Despite on tremendous amount of work which was done in the last decades, it is clear that rule-based and heuristic methods are difficult to use to achieve human-like generated music. And deep learning is not hype but the only method which can achieve a solid result in our days. Nevertheless, it does not mean we should abandon heuristics and rule-based approaches. Recent work (Jaques et al., 2016) shows that applying human knowledge about music can significantly improve results a generated music, but it still works in combination with deep learning models.

We highly encourage to read the survey for a more detailed review of described methods.

2.2 State of the art models

As it was said before, many significant results in machine learning field were achieved using deep learning techniques. One of the most rapidly developing platform Magenta is an art generation project. The project includes many states of the art models for music generation like MusicVAE (Roberts et al., 2018), PerformaceRNN (Oore et al., 2018), COCONET (Huang et al., 2017), Piano Genie (Donahue, Simon, and Dieleman, 2018). We review some of them in details to get a better understanding of why there are successful and which idea they bring.

Nevertheless, it is necessary to notice that there are different models which achieve competitive results.

2.2.1 COCONET

One of the recent successful examples of applying CNN to a music generation task is a model described in "Counterpoint by convolution" paper (Huang et al., 2017). COCONET is a deep convolutional model trained to reconstruct partial scores. The main idea of the work is to fill missed notes in scores, that is why there is a counterpoint - the process of placing notes against notes to construct a polyphonic musical piece. The main reason to try such an approach is the fact that human composers write music in a nonlinear fashion, scribbling motifs here and there, often revisiting choices previously made in order to better approximate this process.

The neural network architecture of COCONET is shown on a Figure 2.1. It uses L stacked convolutional layers and after every second convolution, it introduces a skip connection from the hidden state two levels below to reap the benefits of residual learning. In the paper, the number of layers $L = 64$ and number of channels $H = 128$. After each convolution, there is a batch normalization with statistics tied across time and pitch. Batch normalization rescales activations at each layer to have concrete mean and standard deviation values.

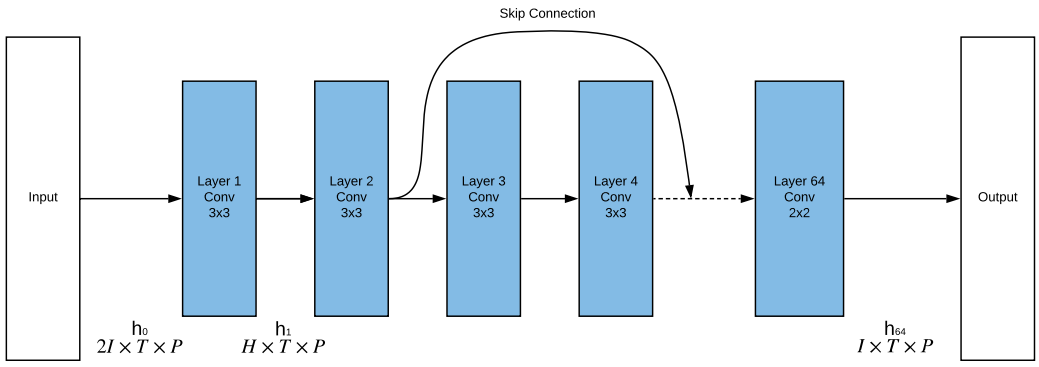

FIGURE 2.1: COCONET architecture

The music is represented as a stack of piano rolls encoded in a binary three-dimensional tensor $x \in \{0,1\}^{I \times T \times P}$. Here I denotes the number of instruments for four-part Bach chorales $I = 4$. T is the number of time steps, P is the number of pitches, and $x_{i,t,p} = 1$ if the i-th instrument plays pitch p at time t. In their work, authors assume each instrument plays exactly one pitch at a time, that $\sum_p x_{i,t,p} = 1$

for all i, t. However, there is no restriction to use more than a pitch at the time, hence present a cord instead of a single pitch.

The most important part of the work is the way the model is training. To train the model is needed to sample a training example x and context C and update neural network weights based on the gradient of the loss. And the place where the sampling quality is a matter. In their work authors compared blocked Gibbs sampling to orderless NADE sampling and provided results where it is shown that blocked Gibbs sampling significantly outperform orderless NADE sampling result.

2.2.2 PerformanceRNN

The idea behind PerformanceRNN model is very intuitive and clear. However, like many deep learning models, there are details which highly affect the final result.

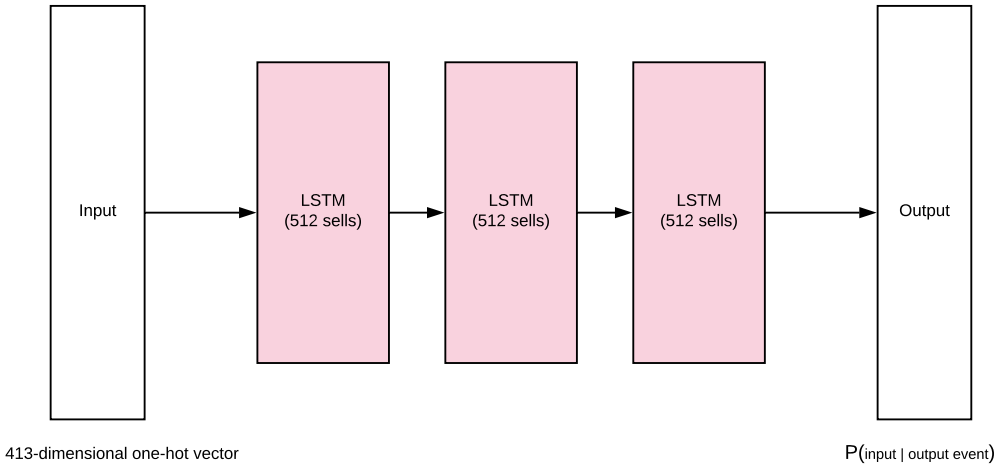

FIGURE 2.2: PerformanceRNN architecture

The neural network architecture is rather simple, it is a stacked 3 LSTM layers and each of them has 512 cells (Figure 2.2). The model operates on a one-hot encoding over this event vocabulary. Thus, at each step, the input to the RNN is a single one-hot 413-dimensional vector. The vector is a representation of the next events:

- 128 NOTE-ON events: one for each of the 128 MIDI pitches. Each one starts a new note.
- 128 NOTE-OFF events: one for each of the 128 MIDI pitches. Each one releases a note.
- 125 TIME-SHIFT events: each one moves the time step forward by increments of 8 ms up to 1 second.
- 32 VELOCITY events: each one changes the velocity applied to all subsequent notes (until the next velocity event).

Note, that a time quantization is done in absolute scale using milliseconds. It makes harder to translate model output to score but at the same time, it makes the output more dynamic and more similar to what people play.

FIGURE 2.3: Events description example (Oore et al., 2018)

Usually, 15 seconds of piano performance contains around 600 of such one-hot-encoded events. Also notice, that time-shift part of the vector restricts note duration to one second. Nevertheless, time shifts can be applied consecutively to allow a longer note duration.

Despite such simple architecture, a described data representation leads to fascinating results which are shown in the paper.

2.2.3 Music Transformer

One of the weakest points for classic sequential neural network models is long-term predictions. Also, music relies heavily on self-reference to build structure and meaning. Frequently, it is self-repetitive and there are multiple recurring elements across a melody line. It reflects in phrases and bigger parts of music compositions such as verse-chorus. To get a coherent piece of music, a model needs to have a broader view of music composition, highlight key elements, repeat and modify them in various ways, create a contrast, tension and release. Self-attention mechanisms are a natural fit for this challenge, as they offer direct access to the generated history, allowing the model to choose the level of detail. So the author of Music Transformer (Huang et al., 2018) decides explore Transformer (Vaswani et al., 2017) architecture and apply it to music generation.

A classic structure of Transformer model (Figure 2.4) has the next important component:

- Positional encoding
- Encoder

- Decoder

Positional encoding is a wave function which aims to resolve an issue of unseen data. For example, the model was trained on sequences of 10 elements maximum it will be difficult to predict 11th element if need. Hence, adding positional encoding helps resolve the issue (what is especially important for music).

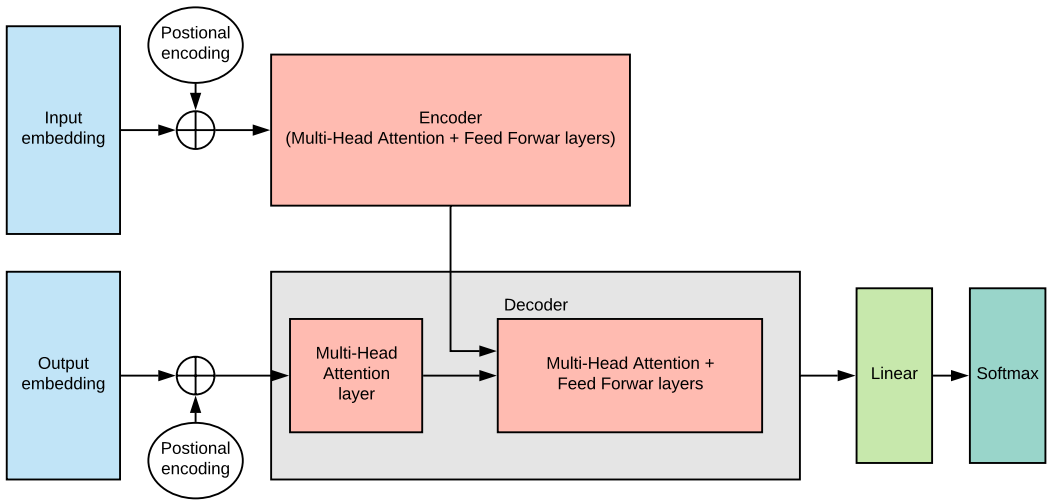

FIGURE 2.4: Transformer architecture

The Encoder is a stack of identical layers and each layer has two sublayers. The first is a multi-head self-attention mechanism (Figure 2.5 right part), and the second is a simple, position-wise fully connected feedforward network.

The Decoder has pretty much the same structure as the encoder. But in addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack.

A classic attention layer transformation has the next pipeline (Figure 2.5 left side). Given vector $x = (x_1, x_2 \dots x_n)$, where $x_i \in \mathbb{R}^{d_x}$. Then an element of attention layer output is:

$$z_i = \sum_{j=1}^{n} a_{ij}(x_j W^v) \tag{2.1}$$

where softmax is:

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{2.2}$$

and scaled dot-product is:

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}} \tag{2.3}$$

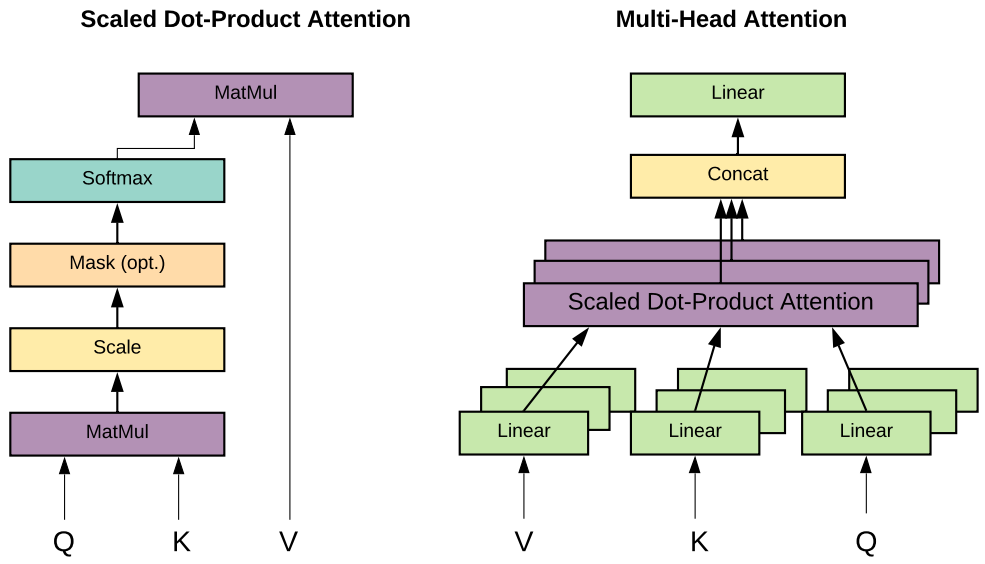

FIGURE 2.5: Transformer attention mechanism

or if to rewrite to a matrix format:

$$z = Attention(Q, K, V) = \frac{1}{\sqrt{d_z}} softmax(QK^T)V \tag{2.4}$$

The model can be improved with adding relative attention (Shaw, Uszkoreit, and Vaswani, 2018)

$$\begin{aligned} z &= RelativeAttention(Q, K, V, R) \\ &= \frac{1}{\sqrt{d_z}} softmax(Q(K^T + R^T))V \end{aligned} \tag{2.5}$$

Authors of Music Transformer took the model with relative attention and extended it to be more suitable for music generation task. They optimized memory consumption for computing relative embeddings and added pitch and timing embeddings to capture more relation information:

$$\begin{aligned} z &= RelativeMusicAttention(Q, K, V, E_r, R_t, R_p) \\ &= \frac{1}{\sqrt{d_z}} softmax(QK^T + Skew(QE_r) + Q(R_p^T + R_t^T))V \end{aligned} \tag{2.6}$$

where:

$Skew(QE_r)$ - optimized relative attention component
R_p, R_t - relative pitch and time embeddings

The improvements allowed outperforms COCONET and PerformanceRNN results and demonstrate a significance of attention mechanism.

2.2.4 Temporal Convolutional Network

Temporal Convolutional Network (Bai, Kolter, and Koltun, 2018) is an attempt to combine all best practice convolutional architectures to a single unit and keep it as simple as possible. The model was not tuned to solve music generation tasks itself. However, the proposed architecture has a bunch of characteristics which allows it to compete for classical sequential models like RNN, LSTM, and GRU in a sequential modeling domain.

The distinguishing characteristics of TCNs are:

- the convolutions in the architecture are causal, meaning that there is no information "leakage" from future to past;
- the architecture can take a sequence of any length and map it to an output sequence of the same length, just as with an RNN but can do it better.

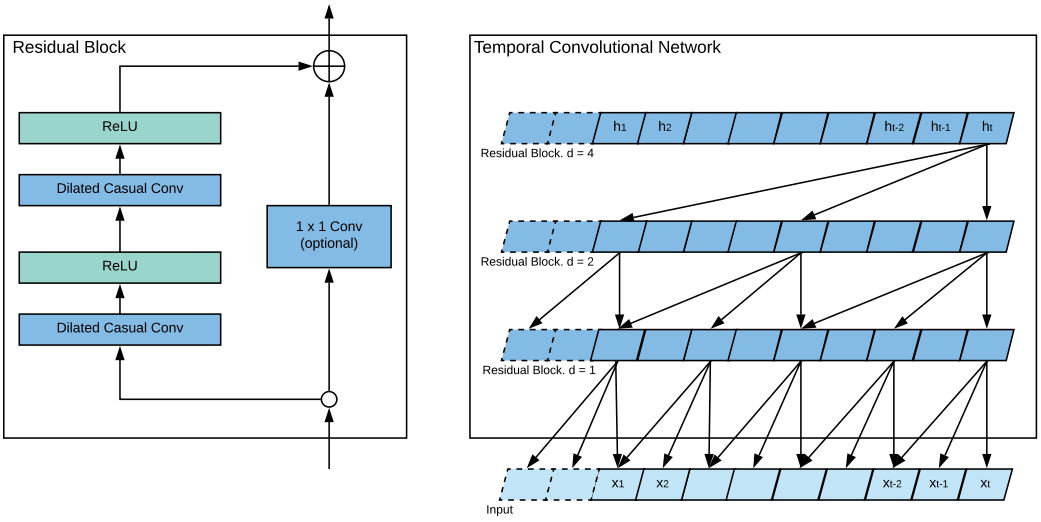

FIGURE 2.6: TCN architecture

The authors of the paper claim that the TCN with virtually no tuning outperforms the recurrent models by a considerable margin. But other models such as the Deep Belief Net LSTM perform better still. Nevertheless, we know that many factors can affect the final result in applying the model in the scope of different problem definition or different training process can lead us to a better result.

Chapter 3

TCN based models

3.1 Default TCN

By "default" we mean the first and the simpler TCN we tested. But in experiments with kernel size it means that a model has three residual blocks.

We consider the next set of hyper-parameters in our research:

- kernel size (convolution layer parameter)
- TCN depth - a number of residual blocks (double dilated conv layers with residual connection). Each block has a dilation coefficient = $2^{(depth-1)}$
- a number of stacks. One stack is a TCN, and 3 stacks is a 3 TCNs connected one-by-one.
- input sequence length

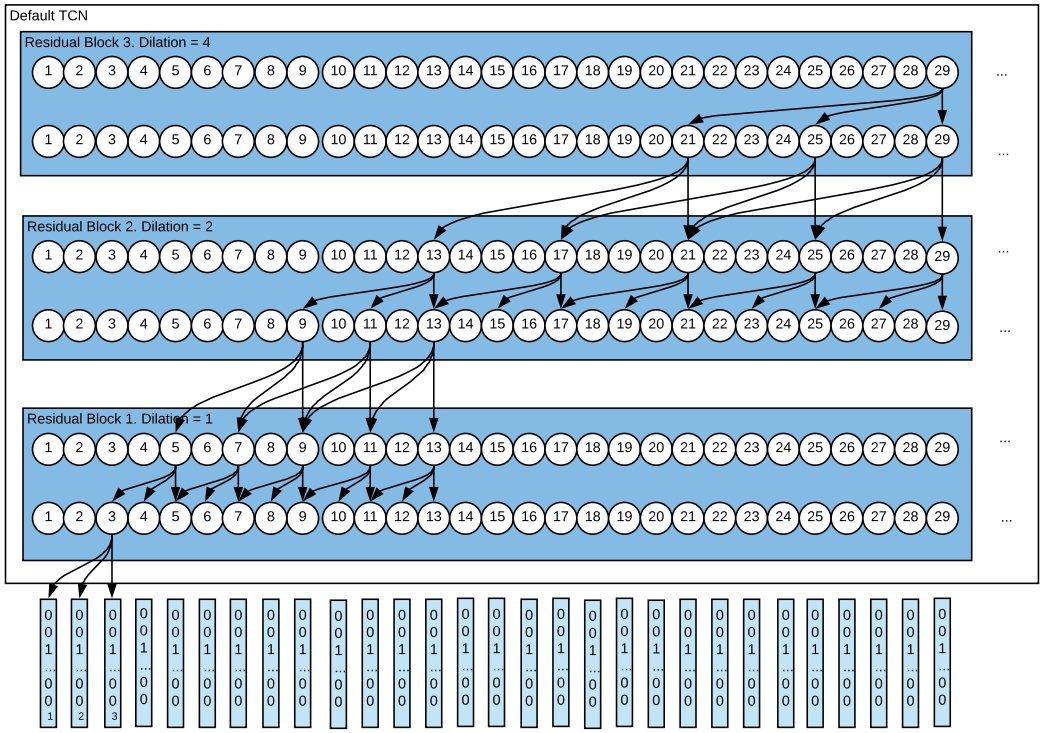

FIGURE 3.1: Default TCN architecture

According to the list, the Default TCN has the next parameters:

- kernel size = 3
- TCN depth = 3. Dilation coefficient = 4
- number of stacks = 1
- input sequence length = 512

The input sequence length and TCN depth is a reflection of PerformanceRNN settings, when kernel size is a minimal value which gives a meaningful results according to TCN authors (Bai, Kolter, and Koltun, 2018). A number of stacks is our hyper-parameter an idea of the parameter will be discussed in Stacked TCN section.

3.2 Stacked TCN

The idea behind the model is to minimize the influence on a prediction of events which are far away for each other. For instance, if we consider a schema on Figure 3.1, we can see that the 1st event in the input affects a prediction of 5th event in the first residual block. If we continue to increase the TCN depth the distance between such event rises exponentially. To decrease the distance growth, we decided to stack TCN with a small number of blocks, in the particular case it is equal to three. In other words, this is a stack of three Default TCN models.

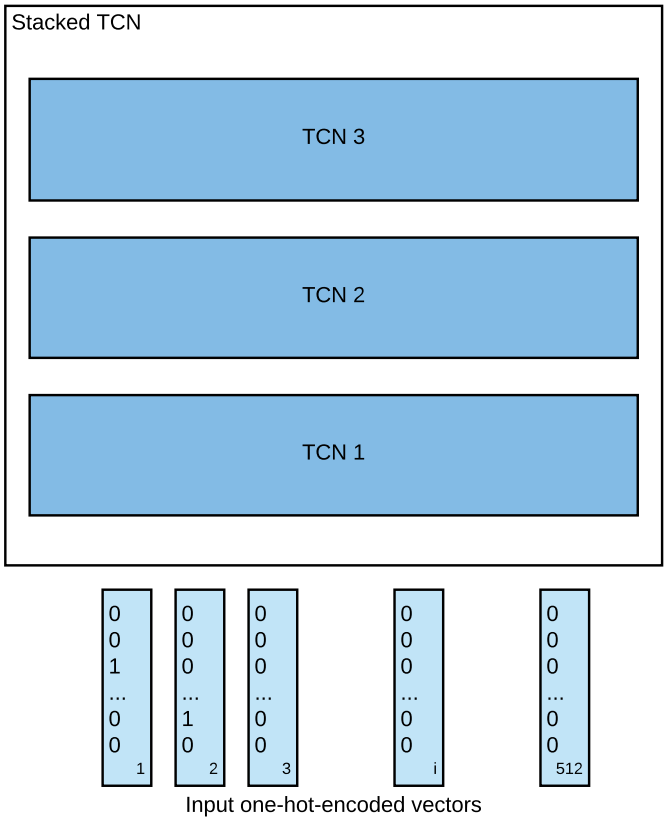

FIGURE 3.2: Stacked TCN architecture

Despite on counter-intuitiveness of the decision in the scope of long-term generation music we expect the model will show better musicality and more "smooth" melody.

3.3 Deep TCN

The idea behind the model is the opposite to Stacked TCN model. Here we want that a prediction on the last event was affected by the earliest event before with the smallest number of the residual blocks.

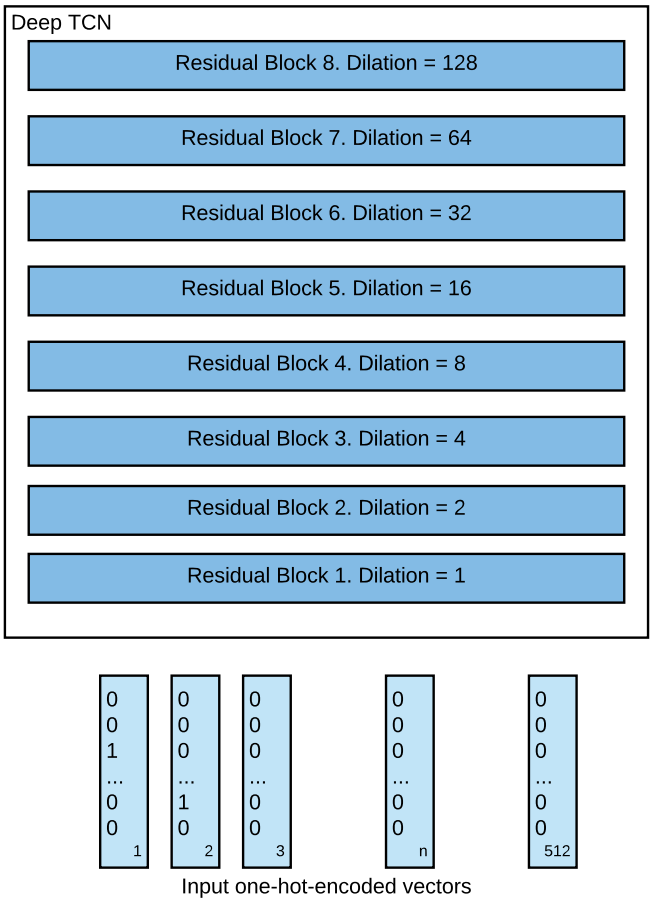

FIGURE 3.3: Deep TCN architecture

In the model with eight blocks, the 1st event affects a prediction of 509th event, which is close to 512 and is good enough in our opinion.

Chapter 4

Experiments

4.1 Setup

4.1.1 Data

MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) is a dataset composed of over 172 hours of piano performances captured on the International Piano-e-Competition. During each installment of the competition, pianists perform on Yamaha Disklaviers which, in addition to being concert-quality acoustic grand pianos, utilize an integrated high-precision MIDI capture and playback system (Hawthorne et al., 2018).

In work, we use the dataset of v.1.0.0. The dataset contains over a week of paired audio and MIDI recordings from nine years of International Piano-e-Competition. The MIDI data includes key strike velocities and sustain pedal positions.

The repertoire is mostly classical, including composers from the 17th to early 20th century. A train/test split configuration is also proposed, so that the same composition, even if performed by multiple contestants, does not appear in multiple subsets.

TABLE 4.1: MAESTRO dataset metadata

Split	Performances	Compositions	Duration (hours)	Note (millions)
Train	954	295	140.1	5.06
Validation	105	60	15.3	0.54
Test	125	75	16.9	0.57
Total	**1184**	**430**	**172.3**	**6.18**

Magenta data pipeline. It is a tool to transform some input data format like MusicXML [1], ABC [2], MIDI [3] to a required output format. Usually, it uses an intermediate step in data transformation, which converts data to a protocol buffer, a flexible and efficient data format.

Our output format is a 512x1 tensor, where each value is an integer between 0 and 387 (388 events). This is a compact format of events representation which was described in PerformanceRNN section. However, during a training process, we convert it back into a one-hot-encoded format. Also, notice that the number of events is 388 and does not match to format which is described in (Oore et al., 2018) (413 events). It is so because of time-shift event; there are 100 events instead of 125,

[1] https://www.musicxml.com/
[2] http://abcnotation.com/about#abc
[3] https://www.midi.org/about

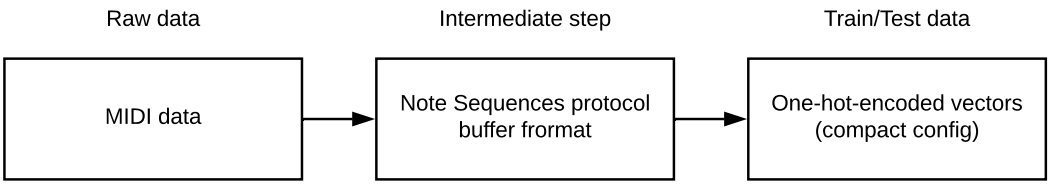

FIGURE 4.1: Data processing pipeline

which means that a time step was changed from 8 ms to 10 ms. It is a recent Magenta team configuration update, and it should not significantly affect experiments.

Data augmentation. It is a common practice in machine learning tasks because it helps improve model quality. For music generation, we use default configurations of Magenta PerformaceRNN data pipeline. It means that:

- Each sample has 5 stretch factors from 0.95 to 1.05 with step 0.25.
- Each sample is transposed up and down all intervals up to a major third.

In the final result, we have 5 samples after sequence stretching (original sample included) and 8 samples after transposition, which gives us 40 samples in total where 39 are new.

4.1.2 Model training process

All models were trained on the same training dataset described in subsection 4.1.1. The training dataset was divided on train and evaluation parts in proportion 90/10. The evaluation part is used for tracking training progress and give a sense of model performance before evaluation on a test dataset.

The models are learned by minimizing a log loss function:

$$logloss = -\frac{1}{N} \cdot \sum_{i=1}^{N} (y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)) \tag{4.1}$$

where:

- N - a number of training examples
- y_i - a probability of a true event
- $\hat{y}_i$ - a probability of a predicted event

It shows the uncertainty of a prediction based on how much it varies from the actual event.

Each convolution layer of TCN model a has 25% of a dropout probability. It is a common practice in machine learning to optimize a training process (Hinton et al., 2012).

All evaluation was done using Amazon EC2 P3 instance with a single Tesla V100 16GB GPU card.

For all model was used Tenserflow framework (version 1.12) on Python3.6.5.

4.1.3 Music generation process

For a music generation, we use 29 performances from a validation split of MAESTRO dataset. We take the first 2-4 seconds (depending on the musical phrase) of each performance and ask models to generate the next N events. After each iteration, we concatenate a generated result with a primal sequence and repeat the generation procedure using the concatenated sequence as an input for the next iteration.

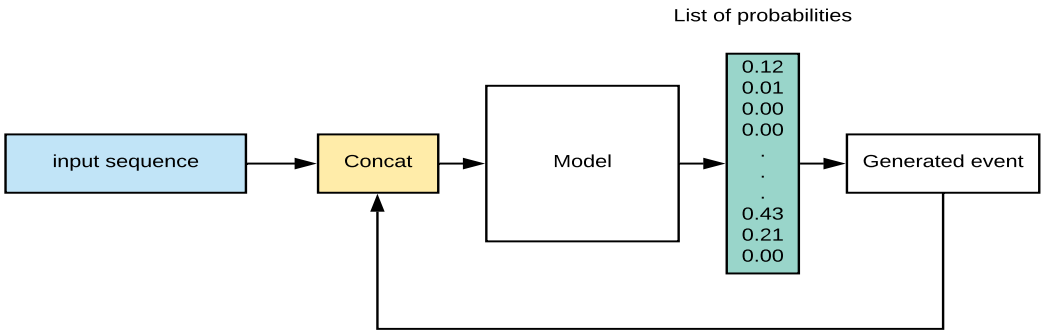

FIGURE 4.2: Music generation pipeline

Models chose an event with a predicted probability, so a generated event is not necessarily a prediction with the highest probability.

4.2 Models comparison

In the section, we compare PerformanceRNN and TCN models training progress using log-loss metric.

Figure 4.3 shows that Default TCN model achieves a similar to PerformanceRNN result and needs less time for that (Table 4.3).

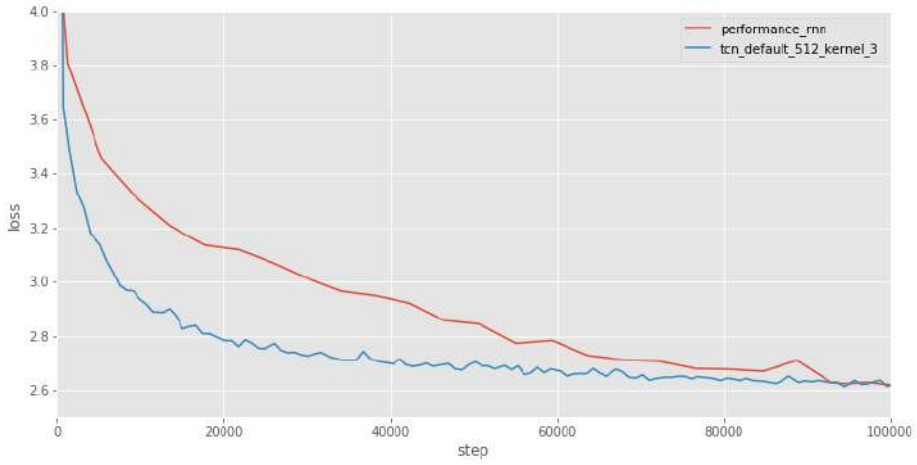

FIGURE 4.3: PerformanceRNN vs Default TCN training progress

More complicated models like Deep TCN and Stacked TCN outperform performance PerformanceRNN and still need less time for training.

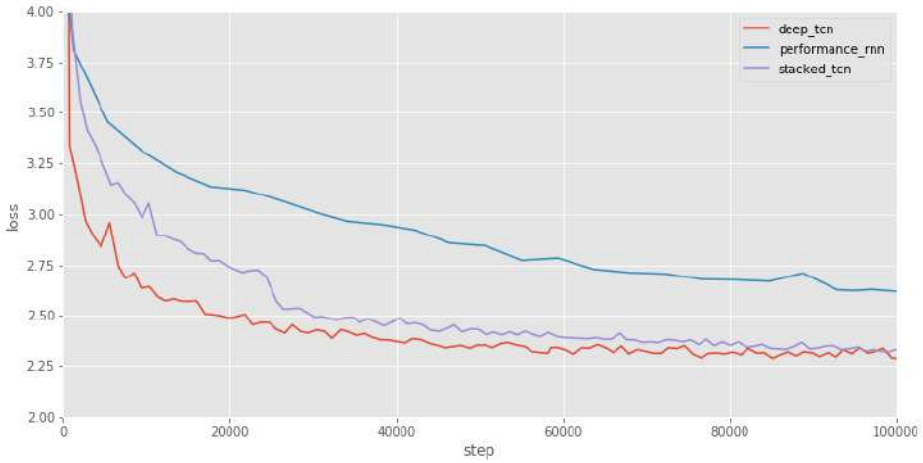

FIGURE 4.4: PerforanceRNN vs Deep TCN vs Stacked TCN training progress

However, all TCN models have a common characteristic, they reached a plateau for very soon, when PerformanceRNN consistently decreases loss metric. It is difficult to predict if PerformanceRNN will outperform the TCN models, but on a short distance, TCN models show a faster convergence.

On a test dataset the best result Stacked TCN. However, it is only slightly better than Deep TCN. Also, log loss metric does not fully reflect generated music quality. For the reason we prepared a human evaluation test.

TABLE 4.2: Models performance on a test data

Model	**Loss**
PerformanceRNN	2.566
Default TCN	2.599
Deep TCN	2.378
Stacked TCN	**2.322**

TABLE 4.3: Models training time

Model	**Aprox. time per 1000 steps (mins)**
PerformanceRNN	14.6
Default TCN	2.5
Stacked TCN	6.26
Deep TCN	5.8

4.3 Human evaluation

Evaluation of a generated music is a non-trivial task which does not have strict set of rules. It requires specific human knowledge which is difficult to describe using mathematical tools.

For the reason, evaluation music by people is the best option in our days. In the experiment, we use Amazon MTurk - an Internet platform for human intelligence tasks (HIT). It allows us to create a listening test and compare samples with each other and evaluate their quality. The study procedure is similar to that is described in the article (Huang et al., 2018). Participants are presented with two musical samples that shared a common priming sequence. Each sample starts with 2 seconds of silence, then a priming sequence was played and then a continuation of that sequence. The continuations were either sampled from one of the models or extracted from a validation set. For each model in the test, we prepared 29 samples [4], 30 seconds each, and create all possible pairs for common priming sequences except the same one. For each task, we required to evaluate a pair of samples by three different people. In result, we got 870 votes which give us the next picture.

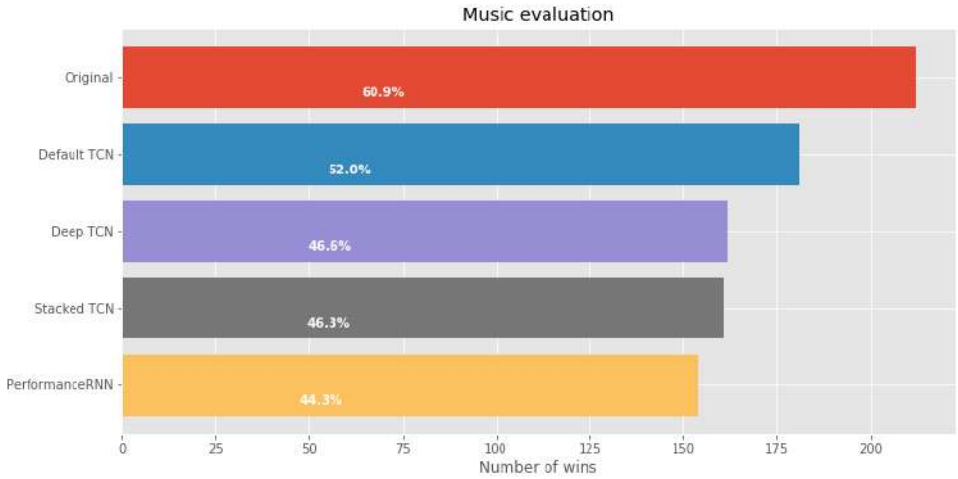

FIGURE 4.5: Human evaluation test

Deep TCN and Stacked TCN show slightly better win rates than PerformanceRNN model when less complicated TCN has a far bigger difference. However, due to technical limitations, it was difficult to compare generated samples with a longer duration. We believe that in such experiment results will have more noticeable differences.

[4] https://clyp.it/user/14wtkg1w - all samples that were used in the experiment are on the. Names for the samples where hashed to prevent a biased evaluation

4.4 Kernel size

For the experiments, there were chosen four different kernel size values (3, 5, 7 and 10) and two different length of sequences (512 and 1024). The idea is to find a value of kernel size when an improvement is not noticeable anymore.

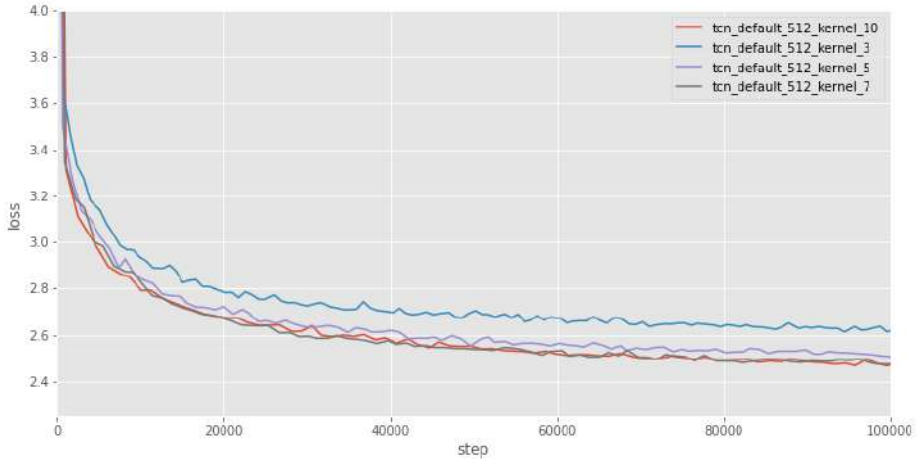

FIGURE 4.6: Default TCN kernel size experiments (for sequence length 512)

For both cases on Figure 4.6 and on Figure 4.7 we can see that there is no a significant improvements for kernel size bigger than five.

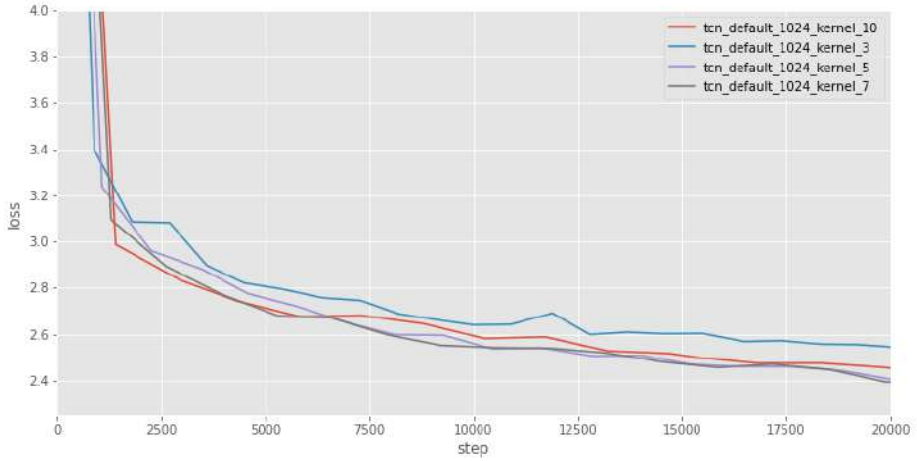

FIGURE 4.7: Default TCN kernel size experiments (for sequence length 1024)

Another observation is that for a longer sequences models show a better training performance (Figure 4.8, the dashed clearly separates results in the end on two groups). And despite our expectations, a model with the biggest kernel size value shows slightly worse performance. However, due to a relatively small number of

training steps, it is hard to judge if it is a model a model overfitting or a random deviation caused by dropout mechanism.

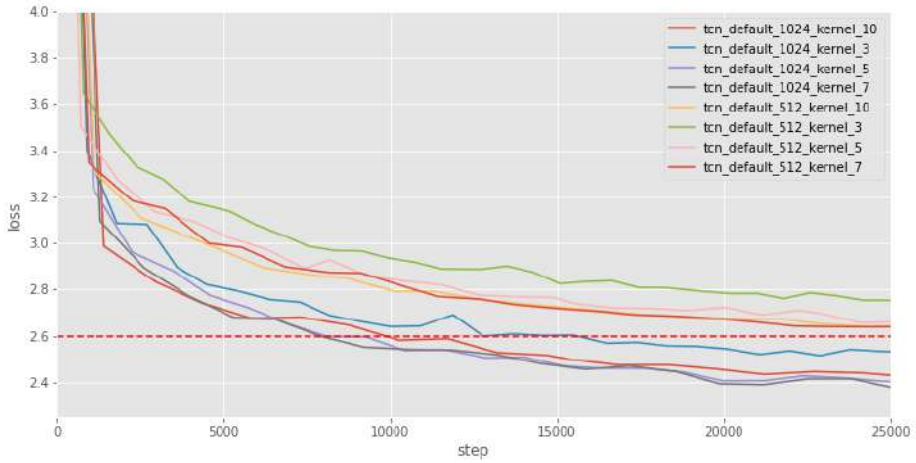

FIGURE 4.8: Kernel experiments comparison (512 vs 1024)

Chapter 5

Conclusion

5.1 Results

In work, we achieved results that are competitive to state of the art models results. Also, we showed that TCN models have a great long term structure perception of music. However, there is still a gap in musicality and consistency of generated compositions. Also, there is a significant boost in decreasing training time in comparison to PerformaceRNN. On the other hand, the models generate music far longer than PerformanceRNN.

The other important thing is that for such creative tasks like music generation it is highly essential to have a human evaluation of the results. It is crucial because such mathematical metrics like accuracy, log loss or perplexity don't reflect the quality of gotten results and require a human expert validation.

5.2 Further work

Improve musicality. There are a few ways to improve musicality of the generated music. For example:

- tune a neural network architecture
- add a rule-based system with a post-correction function

The first approach is about hyper-parameters tuning and exploring different deep learning approaches like changing of an activation function or fine-tuning.

The second approach means that we slightly modify results somewhere in a chain of transformation between model output and writing to a MIDI file.

Speed up a generation process. As it was discussed, a generation process for TCN model is far longer than for a PerformanceRNN. We believe it is because of the inefficiency of developed code and it can be improved. It is important because it will help with models output analysis and speed up further experiments.

Neural network architectures exploring. It has been already shown that the efficiency of TCN is comparable to classic sequential neural network models. However, in work, we explored only basic architectures and didn't try ensembles, concatenation of the models or adding an attention mechanism.

Bibliography

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun (2018). "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: *CoRR* abs/1803.01271. arXiv: 1803.01271. URL: http://arxiv.org/abs/1803.01271.

Donahue, Chris, Ian Simon, and Sander Dieleman (2018). "Piano Genie". In: *CoRR* abs/1810.05246. arXiv: 1810.05246. URL: http://arxiv.org/abs/1810.05246.

Fernández, Jose D. and Francisco J. Vico (2013). "AI Methods in Algorithmic Composition: A Comprehensive Survey". In: *J. Artif. Intell. Res.* 48, pp. 513–582. DOI: 10.1613/jair.3908. URL: https://doi.org/10.1613/jair.3908.

Hawthorne, Curtis et al. (2018). *Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset*.

Herremans, Dorien, Ching-Hua Chuan, and Elaine Chew (2017). "A Functional Taxonomy of Music Generation Systems". In: *ACM Comput. Surv.* 50.5, 69:1–69:30. DOI: 10.1145/3108242. URL: https://doi.org/10.1145/3108242.

Hinton, Geoffrey E. et al. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580. arXiv: 1207.0580. URL: http://arxiv.org/abs/1207.0580.

Huang, Cheng-Zhi Anna et al. (2017). "Counterpoint by Convolution". In: *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pp. 211–218. URL: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/187_Paper.pdf.

Huang, Cheng-Zhi Anna et al. (2018). "An Improved Relative Self-Attention Mechanism for Transformer with Application to Music Generation". In: *CoRR* abs/1809.04281. arXiv: 1809.04281. URL: http://arxiv.org/abs/1809.04281.

Jaques, Natasha et al. (2016). "Tuning Recurrent Neural Networks with Reinforcement Learning". In: *CoRR* abs/1611.02796. arXiv: 1611.02796. URL: http://arxiv.org/abs/1611.02796.

Large, Edward W. and John D. Crawford (2002). "Auditory Temporal Computation: Interval Selectivity Based on Post-Inhibitory Rebound". In: *Journal of Computational Neuroscience* 13.2, pp. 125–142. DOI: 10.1023/A:1020162207511. URL: https://doi.org/10.1023/A:1020162207511.

Lipton, Zachary Chase (2016). "The Mythos of Model Interpretability". In: *CoRR* abs/1606.03490. arXiv: 1606.03490. URL: http://arxiv.org/abs/1606.03490.

London, Justin (2002). "Cognitive Constraints on Metric Systems: Some Observations and Hypotheses". In: *Music Perception Summer* 19.4, 529—550.

Oord, Aäron van den et al. (2016). "WaveNet: A Generative Model for Raw Audio". In: *CoRR* abs/1609.03499. arXiv: 1609.03499. URL: http://arxiv.org/abs/1609.03499.

Oore, Sageev et al. (2018). "This Time with Feeling: Learning Expressive Musical Performance". In: *CoRR* abs/1808.03715. arXiv: 1808.03715. URL: http://arxiv.org/abs/1808.03715.

Ribeiro, Marco Túlio, Sameer Singh, and Carlos Guestrin (2016). ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *CoRR* abs/1602.04938. arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

Roberts, Adam et al. (2018). "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music". In: *CoRR* abs/1803.05428. arXiv: 1803.05428. URL: http://arxiv.org/abs/1803.05428.

Shaw, Peter, Jakob Uszkoreit, and Ashish Vaswani (2018). "Self-Attention with Relative Position Representations". In: *CoRR* abs/1803.02155. arXiv: 1803.02155. URL: http://arxiv.org/abs/1803.02155.

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.