**“AUTOMATED MUSIC GENERATION USING RNN”**

**Major Project-I**

***Submitted by:-***



***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

***in***

**CSE-ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**



**DEPARTMENT OF CSE- ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

GYAN GANGA INSTITUTE OF TECHNOLOGY & SCIENCES

JABALPUR (M.P.)

RAJIV GANDHI PRODYOGIKI VISHWAVIDYALAYA,

BHOPAL (M.P.)

**NOVEMBER- 2023**

#### *CERTIFICATE*

This is to certify that the Minor Project-I entitled “**AUTOMATED MUSIC GENERATION USING RNN**” submitted by **Piyush Nankani, Aradhya Shrivastava, Aniket Pratap Singh and Anushka Malkhede** has been carried out under my guidance & supervision. The project report is approved for submission towards partial fulfillment of the requirement for the award of degree of **BACHELOR OF TECHNOLOGY** in **CSE**-**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** from **RAJIV GANDHI PROUDYOGIKI VISHWA-VIDYALAYA, BHOPAL (M.P).**

| **Dr. Ruchi Patel**  **Project Guide**  **Dept. of CSE-Artificial Intelligence and Machine Learning** | **Dr. Preeti Rai**  **HoD**  **Dept. of CSE-Artificial Intelligence and Machine Learning** |
| --- | --- |

**Dr. Ashok Kumar Verma**

**Dean**

**Dept. of Computer Science and Engineering**

#### *CERTIFICATE*

This is to certify that the Minor Project-I entitled “**AUTOMATED MUSIC GENERATION USING RNN**” is submitted by **Piyush Nankani ,Aradhya Shrivastava ,Aniket Pratap Singh and Anushka Malkhede** for the partial fulfillment of the requirement for the award of degree of **BACHELOR OF TECHNOLOGY** in **CSE-ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** from **RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P).**

Internal Examiner External Examiner

Date: 20th Nov, 2023 Date: 20th Nov, 2023

# 

# 

#### *DECLARATION*

We hereby declare that the project entitled **“AUTOMATED MUSIC GENERATION USING RNN..”** which is being submitted in partial fulfillment of the requirement for award of the Degree of Bachelor of Technology in Computer Science to **“RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)”** is an authentic record of our own work done under the guidance of **Prof. RUCHI PATEL Department of CSE-ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING, GYAN GANGA INSTITUTE OF TECHNOLOGY & SCIENCES, JABALPUR**.

The matter reported in this Project has not been submitted earlier for the award of any other degree.

**Date:** 20th Nov, 2023

**Place: JABALPUR**

#### *ACKNOWLEDGEMENT*

We sincerely express indebtedness to esteemed and revered guide **Dr. RUCHI PATEL of Department of CSE-Artificial Intelligence and Machine Learning** for his/her invaluable guidance, supervision, and encouragement throughout the work. Without his/her kind patronage and guidance, the project would not have taken shape.

We take this opportunity to express a deep sense of gratitude to **Dr. Preeti Rai, Head of Department of Artificial Intelligence and Machine Learning** for her encouragement and kind approval. Also, we thank her in providing the computer lab facility. We would like to express our sincere regards to her for advice and counseling from time to time.

We owe sincere thanks to **Dr. Ashok Kumar Verma, Dean, Computer Science and Engineering** for his encouragement and kind support. We would like to thanks to all the faculties in Department of **CSE-Artificial Intelligence and Machine Learning** for their advice and counseling time to time.



**Date :** 20th Nov, 2023

##### **Place: JABALPUR**

**TABLE OF CONTENTS**



**1 .Introduction**

**2. Motivation**

**3. Problem Statement/ Objective**

**4. Requirement Gathering**

**5. SDLC**

**6. Proposed Solution**

**7. Challenges/Future Plans**

**8. Dataset Collection**

**9 .Technologies used**

**10. Flow Diagram and UML diagrams**

**11. Implemented Screenshots**

**12. Implementation Results**

**13. Code Snippet to the Python Notebook**

**14. Conclusion**

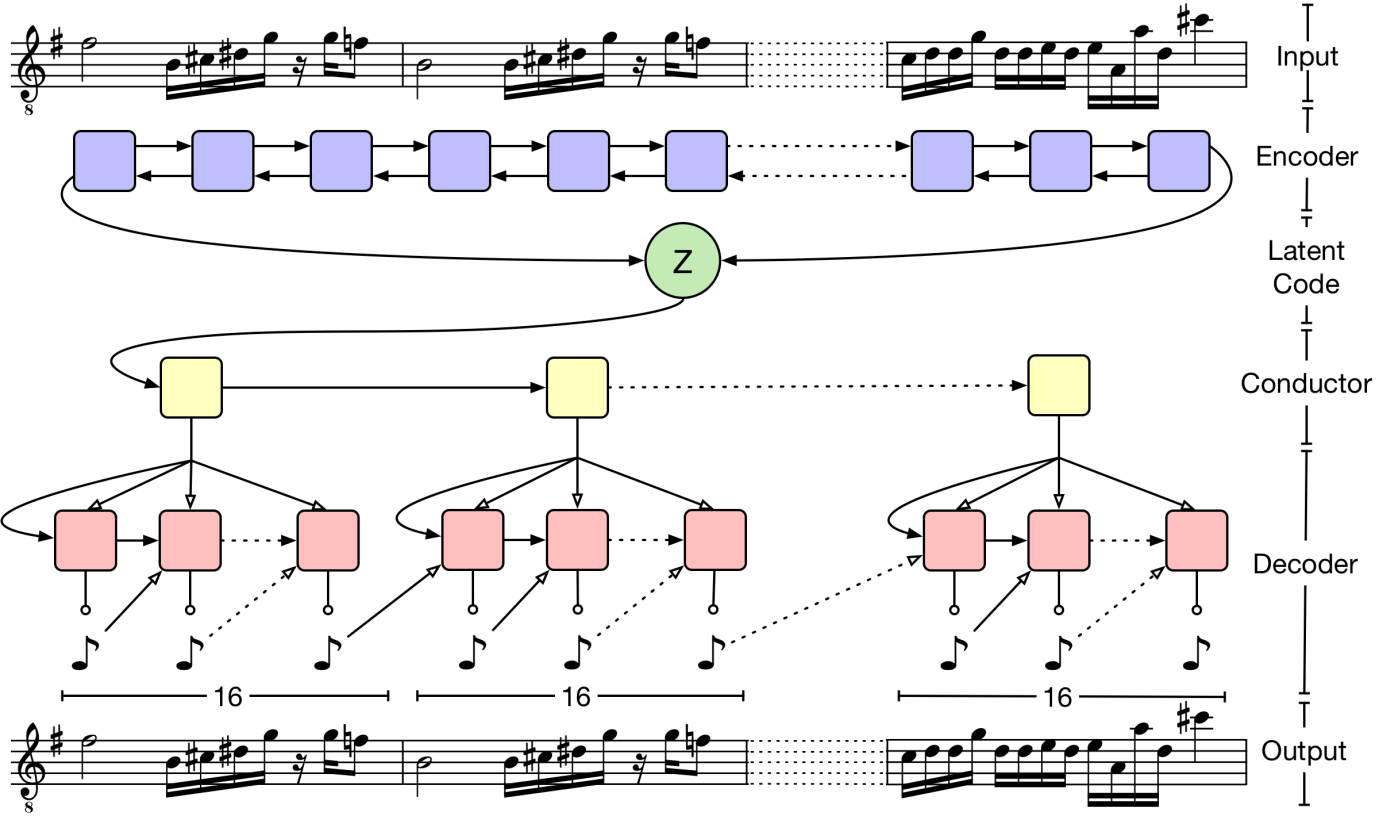
**15. Bibliography**

1. **INTRODUCTION**

Automated Music Generation- This project explores the synergy between artificial intelligence and music composition through the implementation of Recurrent Neural Networks (RNNs). Focused on automated music generation, the RNN model is designed to decipher intricate patterns in diverse musical genres. This report outlines the model's architecture, data preprocessing methods, and evaluation metrics. Beyond technical aspects, it delves into the broader implications of AI in music creation, aiming to spark discussions on the evolving relationship between technology and artistic expression.

Harnessing the power of these smart algorithms, we explore how machines can learn from diverse musical styles and craft their own compositions. RNNs, acting as virtual composers, grasp the intricate patterns in music. In this report, we unveil the architecture of our system, explain how we prepare the musical data, and assess the quality of the generated tunes. Beyond the technical aspects, we ponder the broader implications of AI in the creative world, envisioning a future where technology collaborates harmoniously with human artistry.

Overall, this project is a powerful tool for generating realistic and pleasing music for a wide range of users.



1. **MOTIVATION**

Unlocking the harmony of AI: Exploring the world of music.

Generation through Deep Learning

In a world where artistry and innovation often coverage, our project aims to harness the power of Deep Learning to push the boundaries of music creation.

To explore the motivation behind generating music through AI and how actually things are done.

Bridging Creativity and Technology

Modern AI work is fascinatingly excellent in this field of work, which makes us question “how did it do this?”

**MODERN AI LIKE** –

* MIDJOURNEY AI
* DALL-E 2

**3.1 PROBLEM STATEMENT**

Deep Learning is already becoming extremely prevalent, with purposes relating to computer vision to natural language, medical,voice recognition, generating art, adding sound to silent films, translation software, marketing, and self-driving vehicles. We willapply deep learning networks to producing music. Music does not have a set dimension since it is aseries of sounds and chords.Conventional neural network- based algorithms cannot be used to make music because they presume fixed dimensionality of input and benchmarks and dependence of outputs. As a result, a methodology that learns to map sequences to sequences would have been

beneficial. We intended to express how Recurrent Neural Networks can be used to compose music (RNN).

A Long Short-Term Memory (LSTM) neural network. Whether handmade or synthesised, music production is a notoriouslychallenging feat due to the multiple components. Considering this, we briefly summarise the intuition, theory, and implementationof LSTMs in music generation, createand introduce the network we discovered to achieve this goal, identify and address issuesand challenges encountered, and include potential future improvements for our network.

However, the transition to automated music generation is not without its challenges. Ensuring diversity in musical output remains a critical consideration. The risk of producing formulaic compositions that lack emotional resonance looms large. Striking a balance between the efficiency of AI-driven processes and the nuanced creativity inherent in human composition is pivotal for success.

**3.2 OBJECTIVE**

The objective of this project report is to present a comprehensive exploration of the design, implementation, and evaluation of an automated music generation system utilizing Recurrent Neural Networks (RNN). Focused on leveraging machine learning techniques, the report aims to detail the development process, from data collection and preprocessing to the training and deployment of the RNN model. The objective is to provide insights into the capabilities and limitations of the model in generating diverse and coherent musical compositions. Additionally, the report aims to contribute to the broader understanding of the intersection between artificial intelligence and creative arts in the domain of music composition.

Furthermore, the project seeks to delve into the intricacies of RNN-based music generation, shedding light on the architectural choices, hyperparameter tuning, and model training dynamics. It aims to evaluate the generated musical outputs through robust metrics, gauging aspects such as melodic coherence, harmonic richness, and rhythmic diversity.

Overall, the project's objective encompasses an exploration of user interaction aspects, where interfaces for seed input and parameter adjustments are considered.

1. **REQUIREMENT GATHERING**

**What Kind of models are used for texture Synthesis?**

**Data Source:**

* Diverse Preprocessed training data – MIDI files

**Model Architecture:**

* Recurrent Neural Network used – LSTM and GRU .
* Regularization – Batch Normalisation and Dropout (20%)
* Hyperparameters- unit\_size, batch\_size, and epochs for training.

**Evaluation Criteria:**

* Metrics for assessing music quality - pitch accuracy, rhythm.
* Human Evaluation

**Real-Time Generation:**

* Real-time generation requirements based on user input or requests.
* Determining the acceptable response time for generating music.

**Libraries and Coding language Required for Training:**

* TensorFlow
* Keras
* Numpy
* LSTM, Dense
* Music21
* Scikitlearn

**SYSTEM SEPCIFICATION:**

RECOMMENDED –

* 12GB RAM
* HDD space 20GB
* favorable GPU (6GB)

**5. SOFTWARE DEVELOPMENT LIFECYCLE**

The software development life cycle (SDLC) for texture synthesis involves a series of phases, each with its own set of goals, tasks, and deliverables. The specific steps may vary depending on the project and the development methodology being used, but the following are some general phases that can be applied to texture synthesis:

1. **Planning:** In this phase, the goals and requirements for automated music generation project are defined. This may involve understanding the specific application of the generated chord, the types of input notes/chords that will be used, and any constraints or criteria that need to be met.

2. **Analysis:** This phase involves analyzing the input note(s) and understanding their statistical properties. This may involve techniques such as Fourier analysis, wavelet analysis, or statistical modeling to identify the patterns, colors, and spatial relationships between pixels in the input note(s).

3. **Design:** In this phase, the high-level design of the automated music generation algorithm is created. This may involve selecting the appropriate techniques and algorithms based on the analysis from the previous phase, and designing a system that can generate new chord that meet the desired criteria.

4. **Implementation:** The implementation phase involves writing the code to implement the automated music generation algorithm. This may involve writing code in a specific programming language, such as Python or C++, and using libraries or frameworks that are appropriate for the task.

5. **Testing:** Once the automated music generation algorithm has been implemented, it is tested to ensure that it is generating new notes that meet the desired criteria. This may involve testing with different input textures and comparing the generated notes to the original notes.

6. **Deployment**: After the testing phase is complete, the automated music generation algorithm is deployed into production. This may involve integrating the algorithm into a larger software system, creating a user interface for the algorithm, or making the algorithm available as a standalone tool.

7. **Maintenance:** Finally, the automated music generation algorithm is maintained over time, with bug fixes, updates, and improvements made as needed to ensure that it continues to meet the needs of its users.

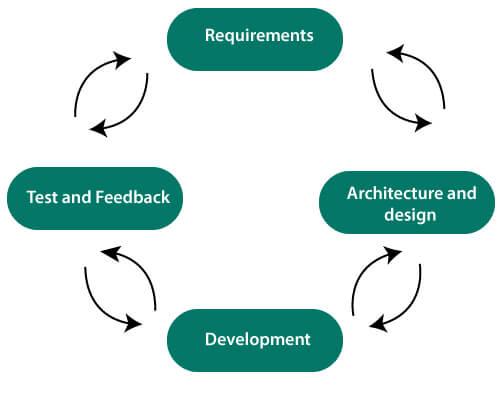
**Overall, the SDLC for automated music generation involves a combination of analytical, design, implementation, testing, and maintenance tasks that are focused on creating a high-quality automated music generation algorithm that meets the needs of its users.**

**WHAT SDLC MODEL IS USED?**

Agile Methodology

(Incremental + Iterative)

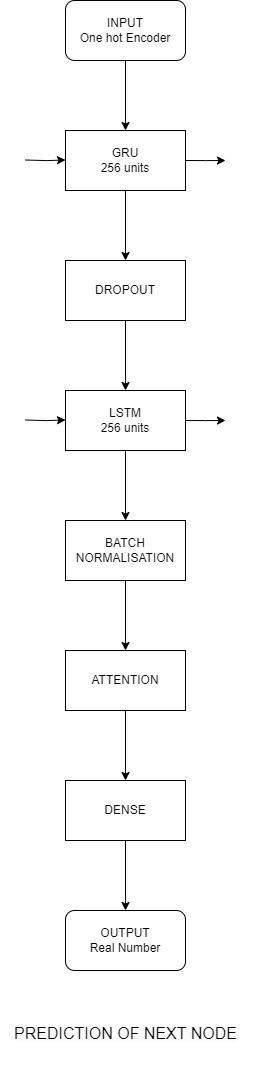
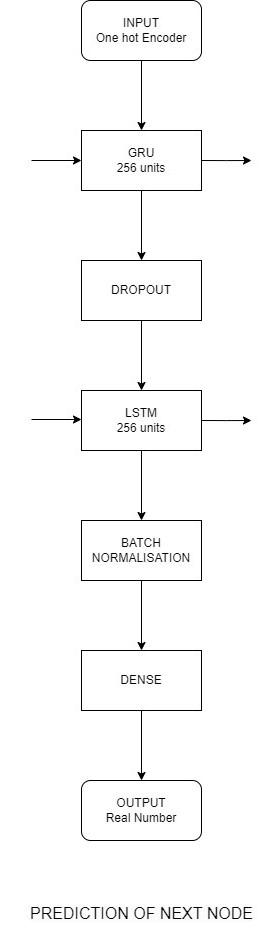
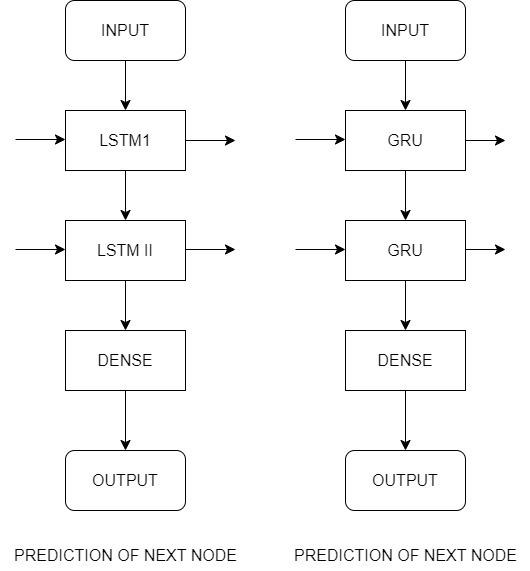
* Agile method combines both incremental and iterative methodology.
* It is iterative because it plans for the work of an iteration to be improved upon in subsequent iterations.
* It is incremental because completed work is delivered throughout the project.



*Fig. 1 Work diagram of Agile Methodology implemented in our project.*

**6. PROPOSED SOLUTION FOR OUR AUTOMATED MUSIC GENERATION**

Deep Learning Recurrent Neural Networks



There are several technologies and techniques that can be used for music generation. Here are some commonly used technologies:

* + TensorFlow or PyTorch: These are powerful deep learning frameworks that offer tools and functionalities for building and training neural networks, including RNNs. They simplify the implementation of intricate models and facilitate the training process.
  + Keras: Keras is an open-source neural network library often used as a high-level interface for building and training machine learning models. It can run on top of TensorFlow or other frameworks, making it convenient for prototyping and experimentation.
  + Musical Data Libraries (MIDI or Audio): Access to musical datasets is crucial for training the RNN model. MIDI (Musical Instrument Digital Interface) or audio data libraries provide the raw material for the algorithm to learn patterns, structures, and styles from existing musical compositions.
  + Music21Library**:** Music21 is a Python toolkit designed for computer-aided musicology. It facilitates the analysis, manipulation, and generation of musical data, making it a valuable resource for projects involving automated music composition
  + Neural networks**:** Neural networks are a type of machine learning algorithm that can be used to generate new textures. They can be trained on a set of input textures and used to generate new textures that are visually similar to the training data.

**Overall, the specific technology used in automated music generation may vary depending on the requirements of the project and the specific application. However, many music generation techniques involve a combination of mathematical analysis, probabilistic modeling, and machine learning algorithms.**

1. **CHALLENGES**

In the process of building and experimenting with this project, we ran into many

challenges, many of which we have overcome, and others which we plan to address in

future work. Among these includes how to evaluate non-deterministic outputs, how the

time between notes should be represented, how the data as a whole should be represented,

and how to have multiple tracks of music generated.

**7.1 Past Challenges**

As has been previously mentioned, music evaluation is subjective and cannot trivially be

assessed without human interaction. While most people can listen to a melody and notice

if there is a note suddenly played out of key, and it would be very reasonable for a

computer to determine this as well, we all still have our own preferences on what sounds

‘good’. When training a model to create music there is no definite way for it to really

judge itself on if the music is good or not, this can make comparing models quite

difficult. On top of this, depending on what sequence we give the models to complete a

single model may produce amazing, or terrible outputs. In order to help mitigate this

issue, we have each model generate a list of outputs to be judged by human evaluators.

In hand-written music, there are plenty of cases where the music may make sudden

changes or jumps with reasoning known only to the songwriter. For instance, to build the

tension a song may repeat the same note many times. These cases would confuse many of

the models we made, and when generating their own music, models would tend to repeat

the same notes forever if they repeated more than a few times.

**7.2 Future Plans**

In its present state, our models do not know how to create beginnings and endings to

pieces, rather, it just starts playing, and ends abruptly at some designated number of

notes. Creating a way for the model to dynamically decide when to end a piece would

allow for non-abrupt endings.

Currently our models use and create single-track songs only. In the future, we will be

allowing our models to create multiple tracks that work together, for instance, it may

produce a melody with a bass to accompany it.

The current input of our model consists of notes and durations for notes. Any chords in

the data set are considered unique notes for our model. This approach is quite limiting

since it

1. Does not differentiate between notes and chords. To give an example, a C major

triad is not seen as a composition of notes, but rather as its own unique note

altogether. Allowing the network to learn how notes can form chords would allow

it to learn a deeper understanding of music.

2) Does not allow some complex compositional patterns. Some examples are

arpeggios, melodies over chords, and meaningful harmony.

In the future, we plan to use the time difference between note on and note off events, very

akin to how MIDI is represented in .mid files. This would allow the model to compose

music more like a human who thinks of chords as the composition of notes.

**8. DATASET COLLECTION**

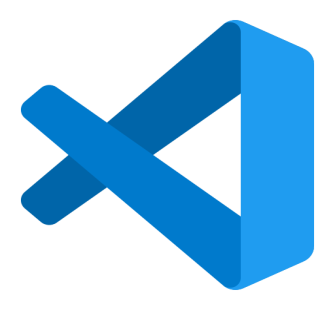
Deep learning models requires a large amount of data to begin with, as they are capable of utilizing GPUs enabling parallel processing. This parallel processing significantly decreases the training time. In our case, the training time was reduced from 25 sec to 10 mins on an average for per epochs.

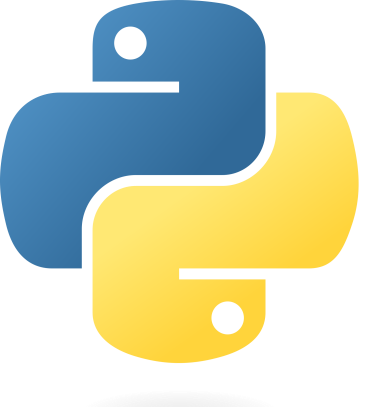
Our dataset has been trained on piano music files in midi format, and these files belong to two broader categories of “classicism” and “romanticism” period.

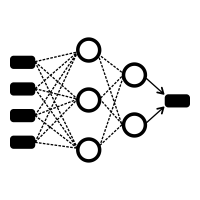
These midi files are then further preprocessed followed by their conversion to the encoding sequence which will then further be processed and used in the model training phase.

**9.TECHNOLOGY USED/STUDY**

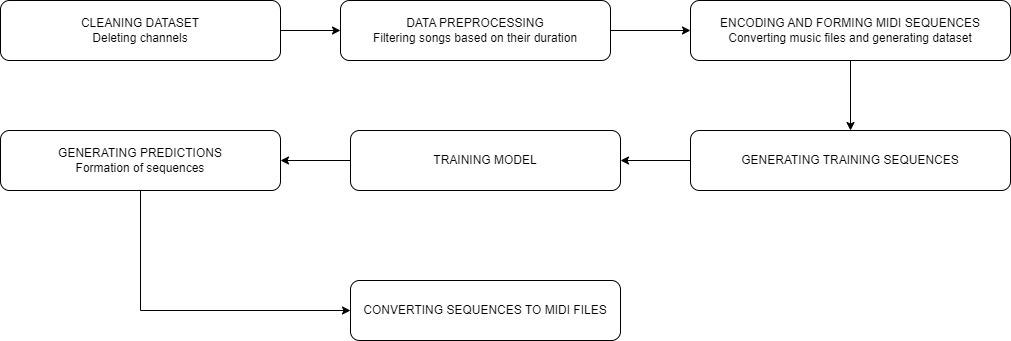
* Deep Learning Libraries
* Convolution Network
* Python
* VS Code





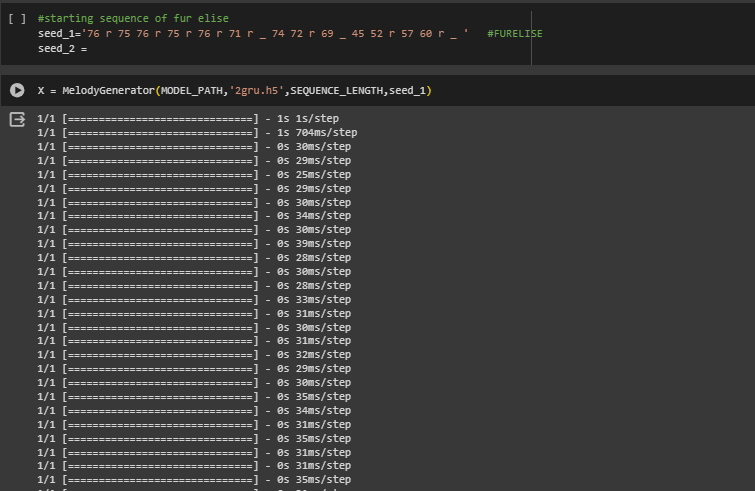


**10. FLOW DIAGRAM**

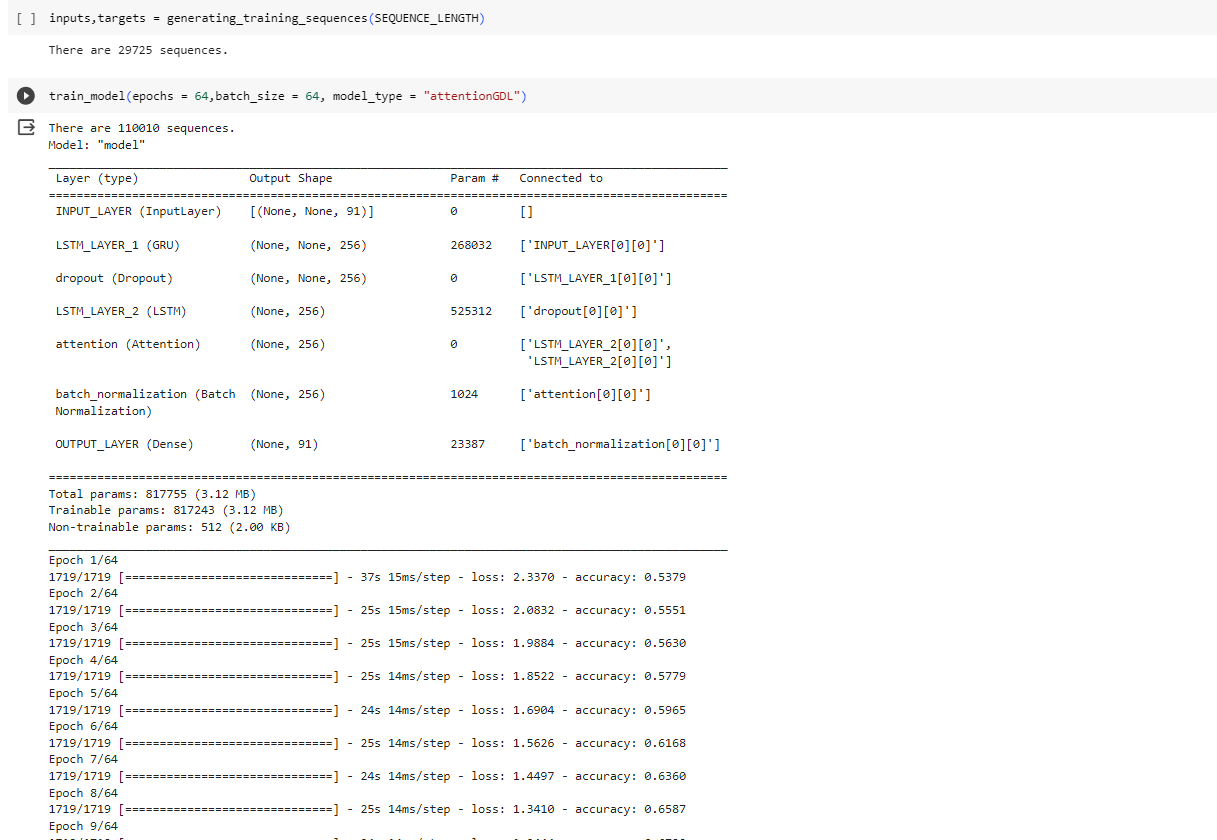


**11. IMPLEMENTED SCREENSHOTS**

MODEL PREDICTION GENERATION:

****

MODEL TRAINING:

****

*Fig. 5 Screenshot of the training process*

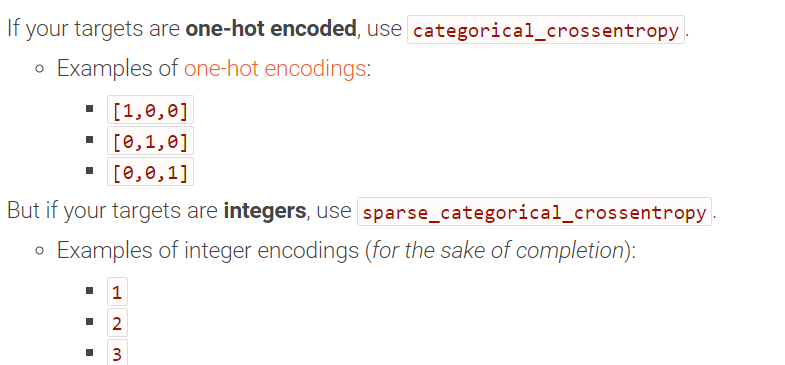
**Using dropout inside the function parameter or outside?**

You can use a Dropout(...) layer, it's not "wrong", but it will possibly drop "timesteps" too! (Unless you set noise\_shape properly or use SpatialDropout1D, which is currently not documented yet)

Maybe you want it, maybe you don't. If you use the parameters in the recurrent layer, you will be applying dropouts only to the other dimensions, without dropping a single step. This seems healthy for recurrent layers, unless you want your network to learn how to deal with sequences containing gaps (this last sentence is a supposal).

Also, with the dropout parameters, you will be really dropping parts of the kernel as the operations are dropped "in every step", while using a separate layer will let your RNN perform non-dropped operations internally, since your dropout will affect only the final output.

CALCULATION OF LOSS FUNCTION:

****

**12. IMPLEMENTATION RESULTS:**

| **Model Name** | **Model Size** | **Epochs, batch size** | **Layer name** | **Loss**  **(in %)** | **Accuracy**  **(in %)** | **Training time** | **Dropout** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2lstm | [256,256] | 50,64 | lstm→lstm→D | 7.2 | 96 | 20 min | 0.3(otf) |
| 2gru | [256,256] | 50,64 | gru→gru→D | 10 | 91 | 18 min | 0.3(otf) |
| \_ | [256,256] | 50,64 | gru→gru | 8.9 | 92.47 | 21 min | - |
| 2lstm\_noD | [256,256] | 50,64 | lstm→lstm | 5.35 | 98.54 | 20 min | - |
| 2lstm\_singleD | [256,128] | 50,64 | lstm→D→lstm | 30 | 90.61 | 16 min | 0.2(itf) |
| mixed\_modelLG | [256,256] | 50,64 | lstm→gru | 6.21 | 97.7 | 18 min | - |
| mixed\_modelGL | [256,256] | 50,64 | gru→lstm | 7.78 | 97.67 | 18 min | - |
| mixed\_modelGLD3 | [256,256] | 50,64 | gru→lstm→D | 13.7 | 94.2 | 19 min | 0.3(otf) |
| mixed\_modelLGD2 | [256,256] | 50,64 | lstm→gru→D | 13.10 | 95.91 | 18 min | 0.2(otf) |
| mixed\_modelLDLB | [256,256] | 50,64 | lstm→D→gru→BN | 9.73 | 96.95 | 20 min | 0.2(otf) |
| mixed\_modelGDGB | [256,256] | 50,64 | gru→D→gru→BN | 17.53 | 94.45 | 18 min | 0.2(otf) |
| mixed\_modelGDLB | [256,256] | 50,64 | gru→D→lstm→BN | 12.79 | 95.91 | 18 min | 0.2(otf) |
| mixed\_modelLDGB | [256,256] | 50,64 | lstm→D→gru→BN | 12.76 | 95.6 | 18 min | 0.2(otf) |
| 2lstm\_doubleD | [256,256] | 50,64 | lstm→lstm | 42.78 | 87.28 | 18 min | [0.2,0.2](itf) |
| **attentionGDLB** | **[256,256]** | **64,64** | **Gru→D→lstm**  **→BN→attention** | **28.95** | **90.98** | **28 min** | **0.3(otf)** |

**otf = Outside the Function**

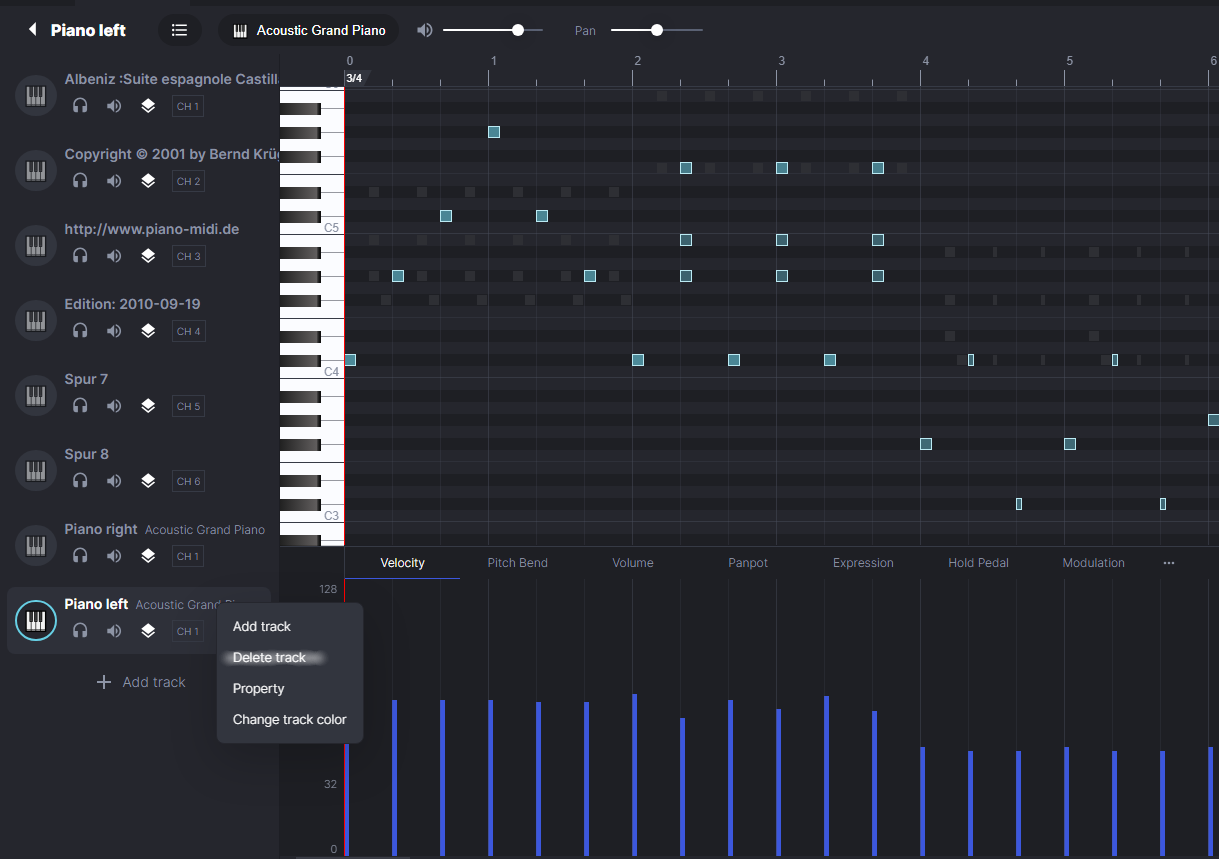
**itf = Inside the function**

**13. CODE SNIPPET TO THE PYTHON NOTEBOOK:**

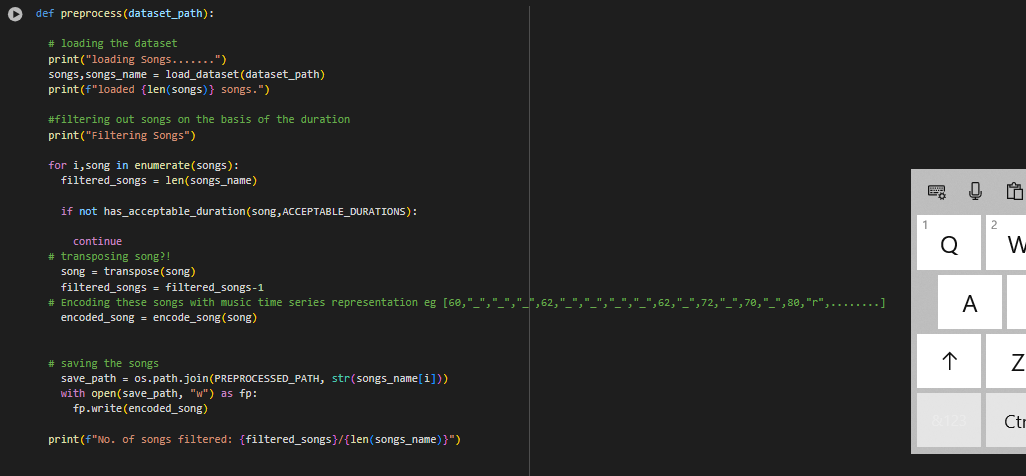
"Automated Music Generation using RNN" - Tensorflow implementation

In summary, we generated chords based on the sample input notes from Classical piano albums of a few artists.

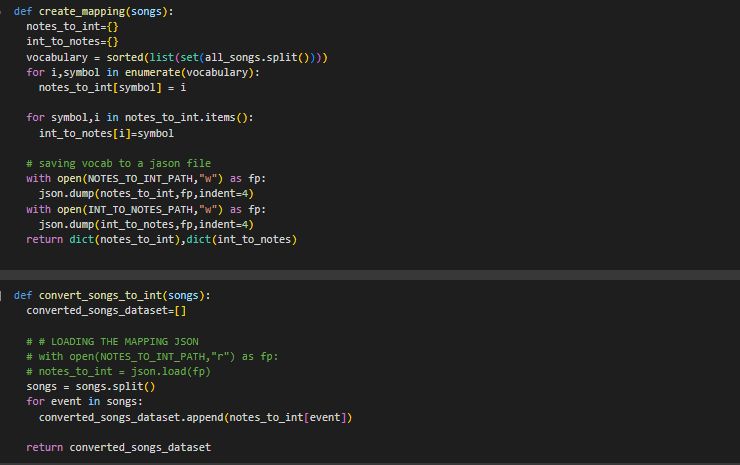
Step 1: Cleaning dataset



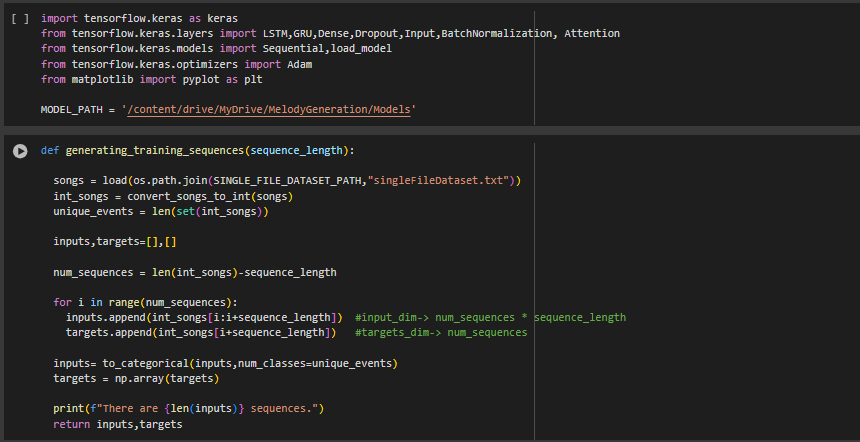
Step 2: Data Preprocessing.



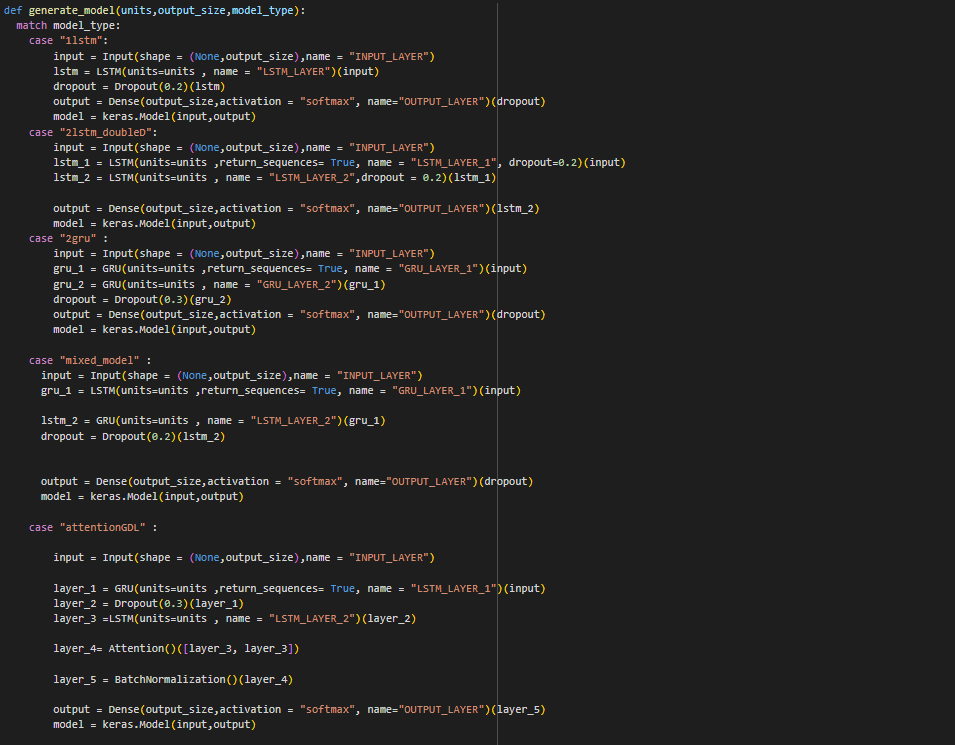
Step 3: Encoding and forming MIDI sequences.



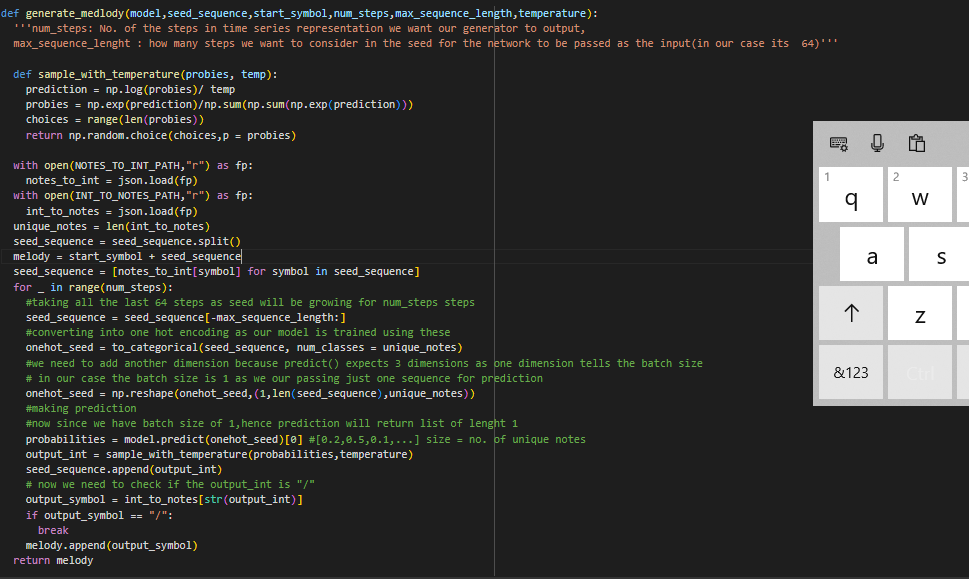
Step 4: Generating Training Sequences.



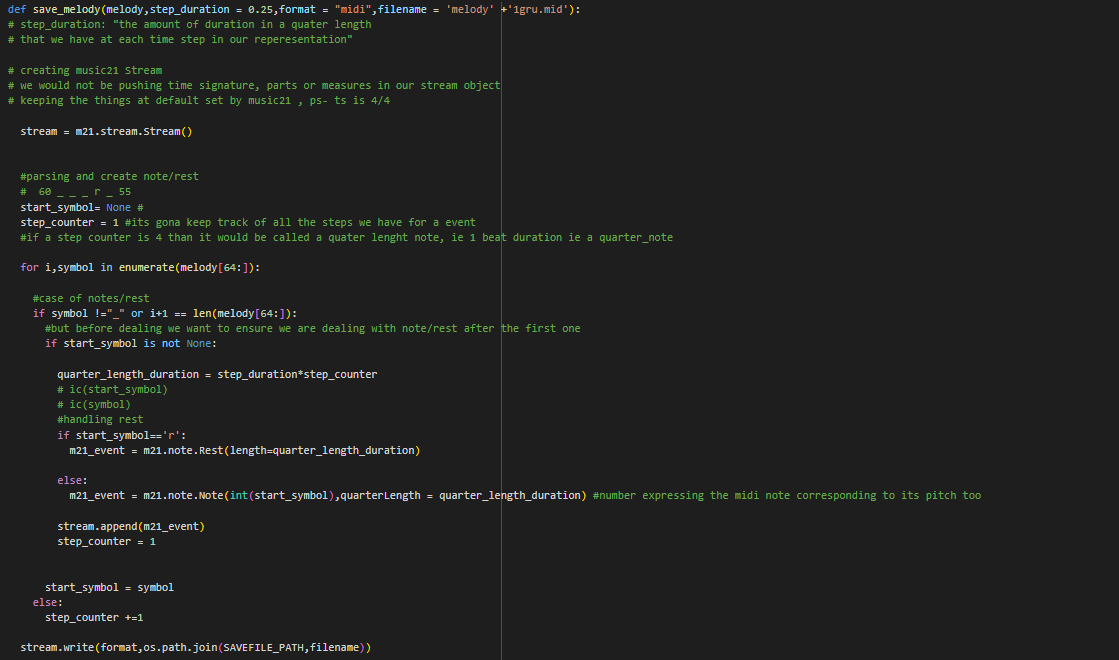
Step 5: Training model, Experimenting architectures.



Step 6: Post-Processing ie. Generating predictions.



Step 7: Converting sequences into MIDI files.



**14. CONCLUSION**

In this report, we gave an overview of the evolution of LSTMs and how we applied them

to music generation. Extending a previous work, we had the data formatted to use one hot

encoding, gave an abundance of different types and genres of music for our network to

Learn on, and, rather than only predicting the notes of a song, we expanded our network to

also predict the durations of notes as well. In the future, we will be expanding our

network to learn and use even more attributes of a song, understand beginning and

endings to songs, and allow for multi-track inputs and outputs.

**15. BIBLIOGRAPHY/REFERENCES**

* https://arxiv.org/ftp/arxiv/papers/2203/2203.12105.pdf

| * arXiv:2203.1215 * https://www.ijraset.com/best-journal/music-generation-using-recurrent-neural-networks |
| --- |