AIP490 Project

FPT UNIVERSITY

# A learning-based method of Automatic Music Transcription

by

**Tai Do, Hoang Nguyen, Hoang Nguyen**

**THE FPT UNIVERSITY HO CHI MINH CITY**

# A learning-based method of Automatic Music Transcription

**by**

**Tai Do, Hoang Nguyen, Hoang Nguyen**

**Supervisor: HaiLT, Ph.D**

A final year capstone project submitted in partial fulfillment of the requirement for the Degree of Bachelor of Artificial Intelligent in Computer Science

**DEPARTMENT OF ITS**

**THE FPT UNIVERSITY HO CHI MINH CITY**

# ACKNOWLEDGMENTS

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report:

1. Le Thanh Hai, Ph.D.:
   - Our supervisor, whose guidance and expertise throughout this project.
   - His insightful feedback significantly improved the quality of our work.
2. We would like to express our deep appreciation to Mr. Nguyen Quoc Trung and Mr. Le Phu Nguyen. Their insightful reviews and constructive feedback provided throughout the project significantly enhanced the quality of our work.

# AUTHOR CONTRIBUTIONS

The contributions of the authors to this work are outlined as follows:

Conceptualization:

- Do Phu Anh Tai: Identified the potential of deep learning for automatic music transcription (AMT) and proposed the research project.
- Nguyen Huu Hoang, Nguyen Duc Hoang: Contributed to refining the project scope and research objectives, focusing on polyphonic piano music transcription.

Methodology:

- Do Phu Anh Tai: Led the research on deep learning architectures for AMT, particularly exploring the use of Long Short-Term Memory (LSTM).
- Nguyen Huu Hoang: Modified the Attention mechanism.
- Nguyen Duc Hoang: Developed the data preprocessing pipeline, including feature extraction, audio segmentation, and music notation labeling.

Writing—Review and Editing:

- Do Phu Anh Tai: Contributed to sections on data preprocessing and neural network architecture selection.
- Nguyen Huu Hoang: Drafted the initial manuscript, mainly responsible for the project's report.
- Nguyen Duc Hoang: Revised the manuscript for clarity and ensured accurate

All Authors have read and agreed to the Final Capstone Project document.

# ABSTRACT

The capability of transcribing music audio into music notation is a fascinating example of human intelligence. It involves perception (analyzing complex auditory scenes), cognition (recognizing musical objects), knowledge representation (forming musical structures), and inference (testing alternative hypotheses). Automatic Music Transcription (AMT), i.e., the design of computational algorithms to convert acoustic music signals into some form of music notation, is a challenging task in signal processing and artificial intelligence. It comprises several subtasks, including multi-pitch estimation (MPE), onset and offset detection, instrument recognition, beat and rhythm tracking, interpretation of expressive timing and dynamics, and score typesetting. This project investigates the application of deep learning for AMT.

**Keywords:** Automatic Music Transcription; Music signals; Music notation

# CONTENTS

# List of Figures

# List of Tables

# 1. INTRODUCTION

Music transcription is the practice of converting music, which was previously un-notated and/or unpopular as written music, into a musical notation. This task is challenging due to the complex nature of music, which can be polyphonic (meaning that multiple notes are played simultaneously) and contain a wide range of dynamics and articulations. This intricate art form bridges the auditory and visual dimensions of music, allowing musicians, composers, and enthusiasts to analyze, interpret, and reproduce musical compositions with precision. Notating a piece or a sound previously unannotated, but needs the knowledge of music to do, with artificial intelligence everyone can do it without learning music theory. The main goal of Automatic Music Transcription (AMT) was to improve the best accuracies that were obtained using traditional Automatic methods and try to get closer to a professional musician's transcription ability.

The primary objective of this project is to explore the capabilities of Deep Learning approaches for tackling the challenge of automatic music transcription. We will delve into various data preprocessing techniques to prepare the raw audio data for neural network models. This stage ensures the data is structured in a way that the models can effectively learn from it. Next, we will experiment with a selection of neural network architectures. Our focus will be on transcribing polyphonic music specifically produced by the piano instrument. By evaluating the performance of different models, we aim to contribute valuable insights into the effectiveness of deep learning for AMT tasks. In our approach, the musical's signal was treated as sequential data and a network model, LSTM (Long Short-Time Memory), was built for realizing audio-to-score conversion. Recordings of piano music in .wav file format is inputted into our algorithm, and the output is MIDI (Musical Instrument Digital Interface) file that can be easily converted into music score.

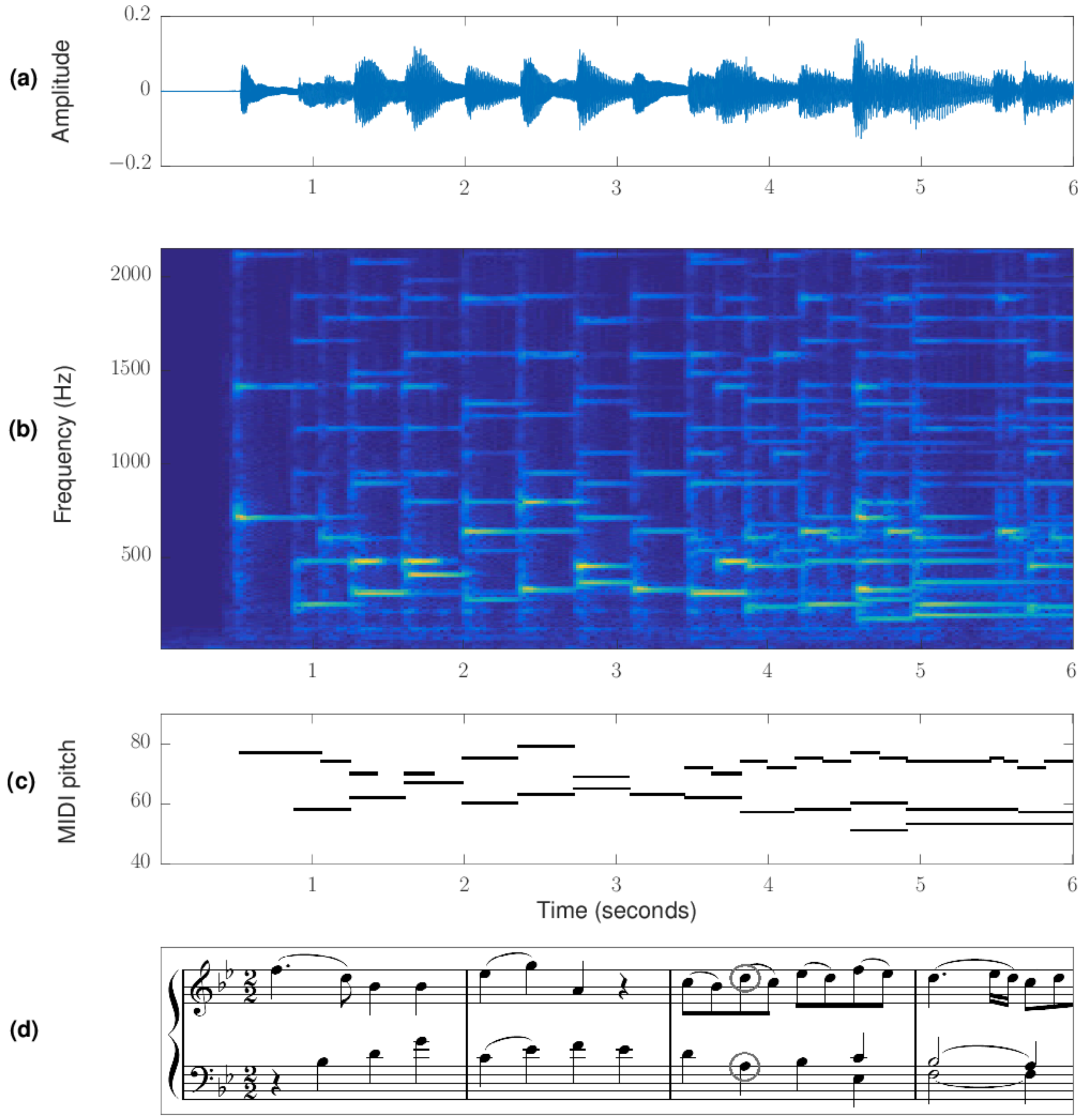


**Figure 1.** Data represented in an AMT system, image from [22]

# 2. RELATED WORK

In the past several decades, multiple attempts have been developed for Automatic Music Transcription (AMT) for polyphonic music. The first approach to AMT was developed in the 1970s by Moorer [1] [2]. Following his work, in the 1990s, Goto and Muraoka [3] achieved decent polyphonic percussion tracks transcription. Unsupervised learning approaches [4] assume that prior knowledge is not required for music transcription, and it's considered as the first step towards Deep Learning approach in Music Transcription. Several approaches include traditional signal processing methods [5], [6], probabilistic modeling [7], Bayesian approaches [8], non-negative matrix factorization (NMF) [9], [10], [11], [12], and neural networks [13], [14]. All of these methods have pros and cons and the research has not converged to a single approach. For example, traditional signal processing methods are simple and fast and generalize better to different instruments, while deep neural network methods generally achieve higher accuracy on specific instruments (e.g., piano). Bayesian approaches provide a comprehensive modeling of the sound generation process, however models can be very complex and slow. With the development of deep learning in recent years, applying neural networks to accomplish AMT has inspired research interests. A model based on CNN was proposed in [15]. A very interesting approach proposed in 2007 [16] used 87 Support Vector Machine (SVM) classifiers to perform frame-level classification, each SVMclassifiers performed an one-versus-all (OVA) operation, and then a Hidden Markov Model (HMM) post-processing was adopted to smooth the results temporally. Various AMT attempts have been developed, however, only a few researches adopted LSTM despite its capability of dealing with sequential data.
Our concepts and codes are adapted based on the work of [17] and [18]. [22] paper also helps to get an overview of Automatic Music Transcription.

## 2.1. Automatic Music Transcription: An Overview

Automatic Music Transcription (AMT)[22] is the task of transcribing music audio into music notation. It is considered a fundamental problem in the fields of music signal processing and music information retrieval (MIR). AMT is closely related to other music signal processing tasks such as audio source separation, which also involves estimation and inference of source signals from mixture observations. It is also useful for many high-level tasks in MIR such as structural segmentation, cover-song detection and assessment of music similarity, since these tasks are much easier to address once the musical notes are known.

However, compared to other problems in the music signal processing field, there are several factors that make AMT particularly challenging:

- **Polyphony:** Music audio contains a mixture of multiple sources (e.g., instruments, vocals) with different pitch, intensity and timbre (sound quality), with each source producing one or more musical voices.
- **Harmonic:** Overlapping sound events often exhibit harmonic relations with each other; for any consonant musical interval, the frequencies form small integer ratios, so that their harmonics overlap in frequency, making the separation of the voices more difficult.
- **Musical Timing:** The timing of musical voices is governed by the regular metrical structure of the music. In particular, musicians pay close attention to the synchronization of onsets and offsets between different voices, which violates the common assumption of statistical independence between sources which otherwise facilitates separation.
- **Annotate:** Annotating ground-truth transcriptions for polyphonic music is very time consuming and requires high expertise.

The paper provided a high-level overview of Automatic Music Transcription, emphasizing the intellectual merits and broader impacts of this topic, and linking AMT to other problems found in the wider field of digital signal processing.

## 2.2. Deep Neural Networks for Piano Music Transcription

Deep Neural Networks for Piano Music Transcription (2017)[17] is a project developed by Diego González Morín. The project likely focuses on applying deep learning techniques, specifically comparing Deep Neural Networks (DNN)  and Long Short-Term Memory (LSTM) Networks performances, to the task of Automatic Music Transcription (AMT) for piano music. The project had concluded that DNN with 3 layers and a Dropout rate of 10% (0.1) got the best performance.

## 2.3. Music Transcription Using Deep Learning

Music Transcription Using Deep Learning (2017)[19] developed by a team from Stanford University: Luoqi Li, Isabella Ni, Liang Yang. Inspired by [17], the paper inherited the [17] model but did some modifications as well as verified the performance between Deep Neural Networks (DNN)  and Long Short-Term Memory (LSTM) Networks in Automatic Music Transcription (AMT) tasks for piano music.

# 3. PROJECT MANAGEMENT PLAN

**Table 1.** Project plan

| Task name | Priority | Owner | Start date | End date | Status | Issues |
|---|---|---|---|---|---|---|
| Find documents | High | All Students | … | … | Done | … |
| Review papers | High | All Students | … | … | Done | … |
| Review and analyze dataset | High | Duc Hoang | … | … | Done | … |
| Experiment | Medium | All Students | … | … | Done | … |
| Compare results | Medium | Tai | … | … | Done | … |
| Writing appendix | Medium | Huu Hoang | … | … | Done | … |
| Future work | Low | All Students | … | … | Defined | … |

**Table 2.** Source code and data

| Items | Link | Description |
|---|---|---|
| Data 1 | MAPS | … |
| Data 2 | MAESTRO | Version 3.0.0 |
| Source code | Link | … |

# 4. MATERIALS AND METHODS

## 4.1. Dataset:

### 4.1.1. Dataset Description:

To accomplish the proposed goal of the project, the selection of the dataset was as important as the Neural Network structures chosen. In this case, we carried out our experiments using the publicly available dataset: ”MIDI Aligned Piano Sounds (MAPS)” [24] and "MIDI and Audio Edited for Synchronous Tracks and Organization (MAESTRO)" [25]. The main advantage of the two datasets is that every WAV file comes along with a text file with different pitch durations and exact onset time.
MAPS contains a diverse collection of piano sounds, including isolated notes, chords, and complete pieces. Notably, each audio recording is meticulously aligned with a corresponding text file detailing the pitch, duration, and exact onset time of each note. This precise alignment between audio and note-level information makes MAPS a valuable asset for training models to handle musical tasks. More detailed statistics on the MAPS dataset are available at https://adasp.telecom-paris.fr/resources/2010-07-08-maps-database/.
The MAESTRO dataset, on the other hand, consists of over 200 hours of paired audio and MIDI recordings from ten years of International Piano-e-Competition. Audio and MIDI files are aligned with $\simeq$ 3 ms accuracy and sliced to individual musical pieces, which are annotated with composer, title, and year of performance. Uncompressed audio is of CD quality or higher (44.1–48 kHz 16-bit PCM stereo). More detailed statistics on the MAESTRO dataset are available at https://magenta.tensorflow.org/datasets/maestro.

### 4.1.2. Dataset Preprocessing:

The first step was to divide the dataset into 3 subsets for training, validation and testing set based on 80-10-10 percentage ratio.

| Dataset | Train | Val | Test |
|---|---|---|---|
| MAESTRO | 962 | 174 | |
| MAPS | 23,928 | 2,991 | 2,991 |

**Table 3.** Train, Valid and Test set

Since it is inefficient to deal with audio files directly, to extract features, the spectrum of the audio signal '.wav' files is used as the main feature of the system. Methods widely used for preprocessing audio signals include Short-Time Fourier Transform (STFT), Mel Frequency Cepstrum Coefficients (MFCC) and Constant Q Transform (CQT) [21]. Upon basic research, MFCC is a feature extraction technique widely used in speech and audio processing and has been used for various machine learning tasks, such as speech recognition and music analysis. Holding a strong belief in the potential of MFCC, it was fairly disappointing to discover that CQT outperforms MFCC in music notating tasks based on the experiments in [17]. Therefore, CQT is chosen as the preprocessing method to extract features for our model.

Inherited from the experiments in [17], firstly, audio signals and a sampling frequency of 44.1kHz were extracted from the '.wav' file. The audio signals were then transformed first from stereo to mono by computing the mean of the multichannel signal, and then Constant Q Transform is separately extracted from the mono audio signal. The CQTs were computed over 7 octaves **(C, D, E, F, G, A, and B**, illustrated in **Figure 2)** with 36 bins per octave and a hop size of 512 samples. Consequently, 252 features per frame were obtained.

F
G
A
B
C
D
E
F
G
A
B
C
D
E

**Figure 2.** Piano Note, image from "https://www.pianote.com/blog/how-to-read-piano-notes/"

The data was represented as a matrix of dimensions number of frames x number of features. The frames of each song were concatenated together and split into individual files of 40000 frames per file.

## 4.1.3. Labels:

As each audio file had a text file with information about the pitch, duration and onset time, time pairing the pitches with each frame was straightforward. The labels are created from the '.txt' files in the form of multilabel one-hot representation. Table 3 shows a small part of a text file. The number of possible pitches was 88, and therefore, the label for each frame was represented as a vector of dimension 88, setting to one of the pitches that were played in the current frame and zero otherwise. This means, at each time frame, a vector of 88 elements corresponds to the 88 notes in a piano, with 1 representing that a note is played at the time frame and 0 representing that a note is not played. The vectors were stacked together and separated into files of 40000 thousand frames per file. This way, each feature file has its corresponding paired labels file to align with the input matrix.

**Table 4.** Text file example. It contains onset/offset time and notes

| OnsetTime | OffsetTime | MidiPitch |
|---|---|---|
| 0.988281 | 1.111979 | 34 |
| 1.109375 | 1.135417 | 53 |
| 1.109375 | 1.161458 | 62 |
| 1.153646 | 1.196615 | 58 |
| 1.518229 | 1.549479 | 74 |
| 1.527344 | 1.563802 | 62 |
| 1.533854 | 1.566406 | 58 |
| 1.539062 | 1.566406 | 82 |
| 1.518229 | 1.567708 | 86 |
| 1.540365 | 1.578125 | 53 |
| 1.518229 | 1.602865 | 77 |
| 1.652344 | 1.682292 | 82 |
| 1.634115 | 1.705729 | 74 |
| 1.628906 | 1.705729 | 86 |
| 1.651042 | 1.746094 | 62 |
| 1.649740 | 1.760417 | 58 |
| 1.661458 | 1.776042 | 53 |
| 1.923177 | 1.949219 | 74 |
| 1.917969 | 1.958333 | 87 |

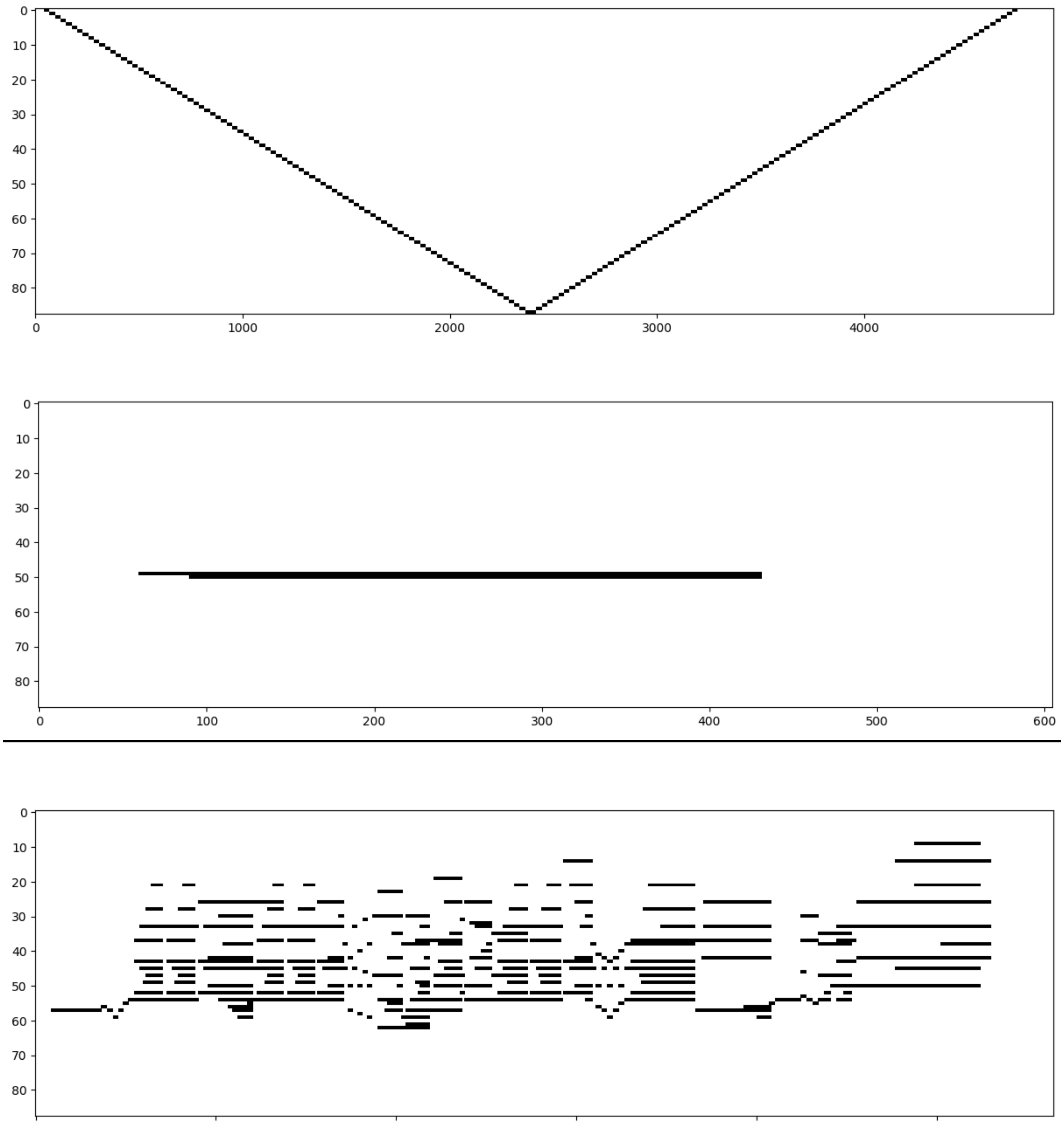


**Figure 3**. Labels Examples

## 4.1.4. Data Insight:

First insight into the data, our team recognised that there could be an imbalance between the label 0 and label 1 since a normal person can only play a maximum of 10 notes at a time and there are 88 notes in a standard piano. By plotting the Labels data (**Figure 3**), we can verify our assumption. Our team did a simple comparison by counting the total number of Label 0 and Label 1 in the training set. The difference between Label 0 and Label 1 is ~41 times (5.36B/0.129B).

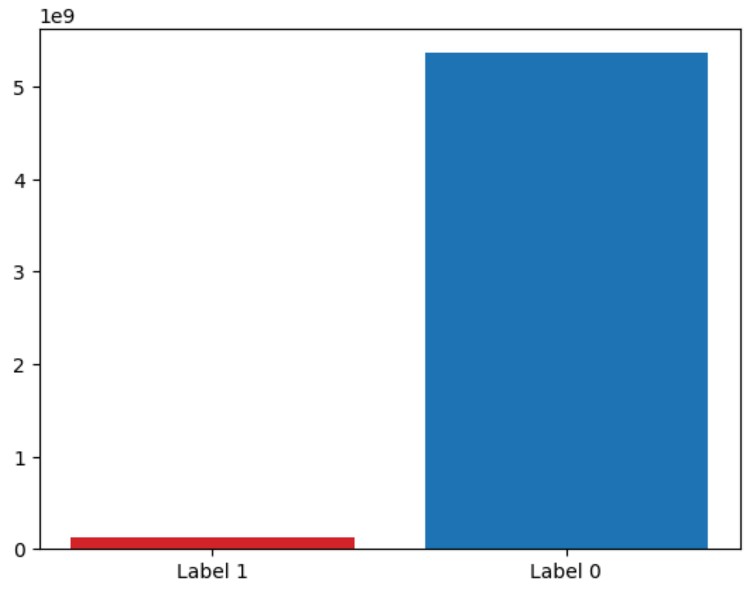


**Figure 4**. Total number of Label 1 vs Label 0

Due to imbalanced data, the Machine Learning model might be exposed to the majority class much more frequently, creating bias toward the majority class, in this case is label 0. To address the problem our team came to a solution of using Weighted Binary Cross-Entropy (an alternative of Binary Cross-Entropy) which will be later explained in **section 4.2.11**.

## 4.2. Method:

As mentioned in the previous **section 4.1.2** Constant Q Transform (CQT) is used for audio feature extraction. Due to time constraints, this project will focus on one main type of Networks: Long Short-Term Memory (LSTM), and several mechanisms: Bidirectional, Attention, Multihead Attention.

### 4.2.1. Constant Q Transform:

The Constant Q Transform (CQT) [21] is a technique that transforms a time-domain signal into the time-frequency domain so that the center frequencies of the frequency bins are geometrically spaced and their Q-factors are all equal. CQT is closely related to the Discrete Fourier Transform (DFT), but with output bins spaced logarithmically in frequency, rather than linearly. A standard DFT uses a constant window size throughout all frequencies. This typically leads to a pretty consistent, fully continuous transformation. However, the constant bin size for all frequencies leads to some problems when you map frequency on a logarithmic scale. Specifically, peaks on the lower end are incredibly wide (sometimes up to half an octave), lacking any sort of detail.

CQT solves this problem by increasing the window size for lower frequencies, and alleviating some of the computational strain caused by this by reducing the window size used for high frequencies. CQT is defined by:

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[K]-1} W[k,n]x[n]e^{\frac{-j2\pi Qn}{N[k]}}$$

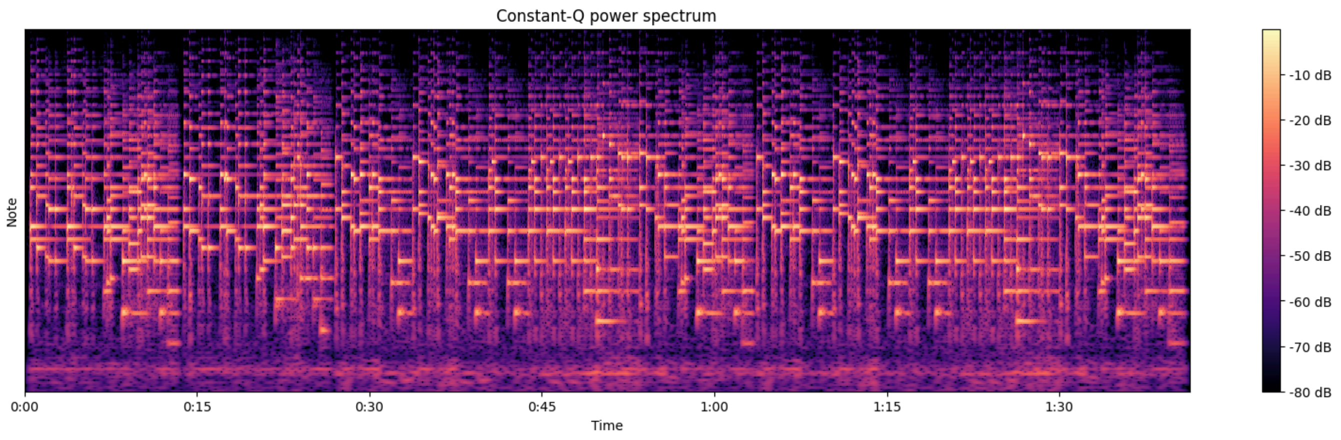


**Figure 5**. CQT Spectrum

## 4.2.2. Long Short-Term Memory:

Humans don't start their thinking from scratch every second. When reading an essay, one person understands each word based on their understanding of previous words. We don't throw everything away and start thinking from scratch again. Our thoughts have persistence.
Recurrent Neural Networks (RNN) [27] have been widely applied to speech recognition and handwriting recognition due to their capability to deal with sequential data. All recurrent neural networks have the form of a chain of repeating modules of neural networks. In standard RNN, this repeating module will have a very simple structure, such as a single tanh layer.

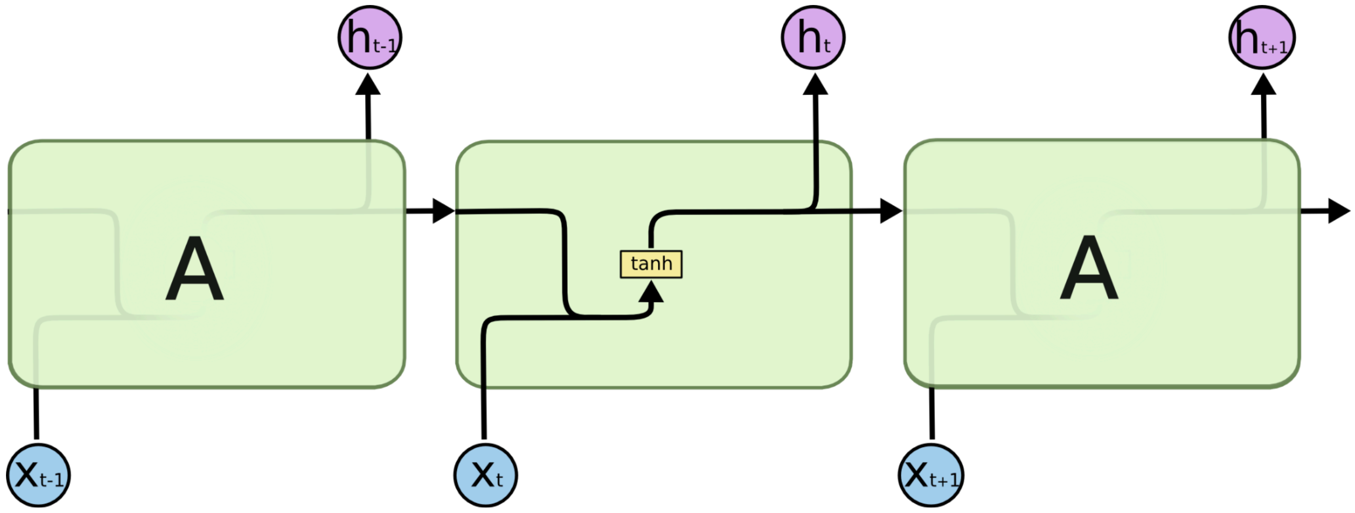


**Figure 6. RNN Cell**, image from "https://colah.github.io/posts/2015-08-Understanding-LSTMs/"

However, since only one past unit is included in the frame of RNN, vanishing gradients during backpropagation become the major problem.

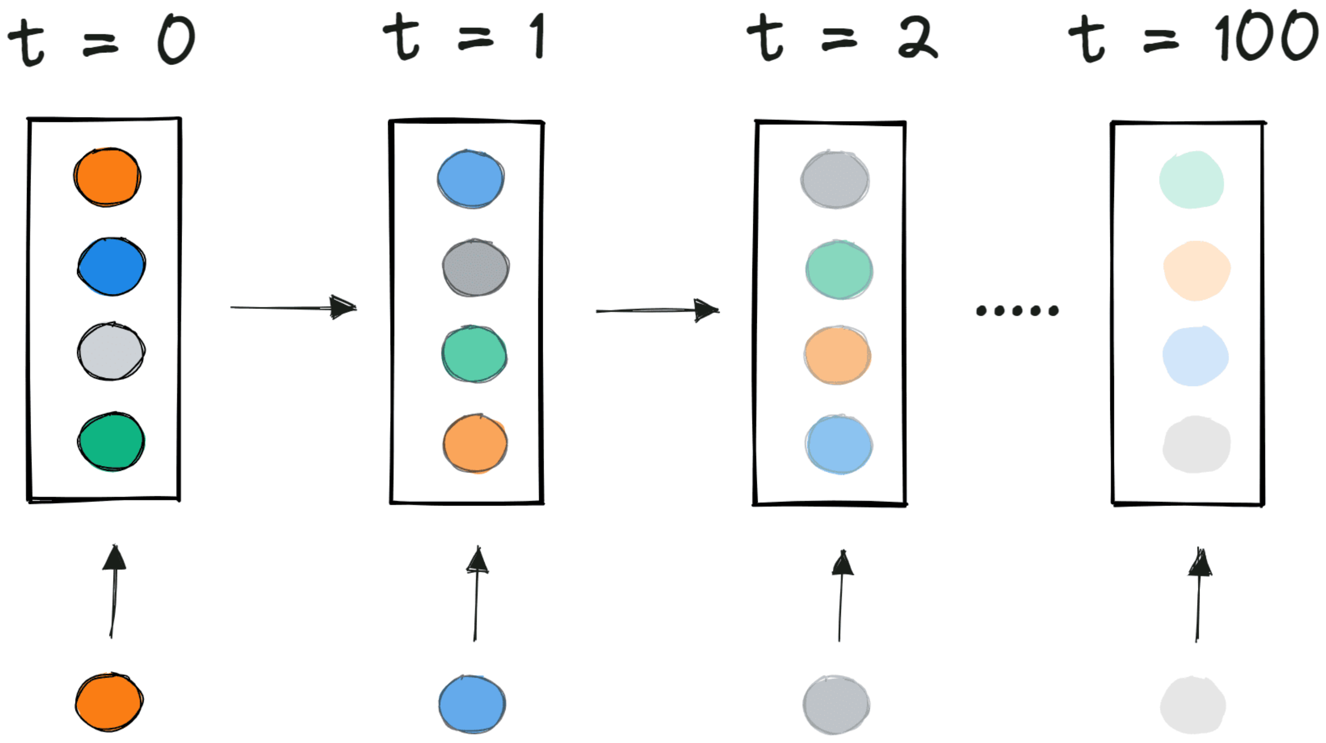


**Figure 7.** Vanishing Gradient, image from "https://www.kdnuggets.com/2022/02/vanishing-gradient-problem.html"

One possible solution is a special kind of Recurrent Neural Network (RNN) called Long Short-Term Memory Networks (LSTM) [26]. Unlike traditional RNN that struggle to capture long-term dependencies due to the vanishing gradients problem, LSTM possesses a unique architecture specifically designed to overcome this limitation. LSTM architecture incorporates several key components:

- Cell State: LSTM introduces the cell state, which acts as a kind of internal memory. Unlike in RNN, the cell state represents the memory of the network, storing information over time, while the hidden state contains the information that is passed to the next time step. Allowing the network to learn long-range dependencies within sequences.

- Gates: LSTM utilizes specialized gates that regulate the flow of information within the network. These gates, namely the forget gate, input gate, and output gate, control what information is retained from the cell state, what new information is added, and what information is exposed to the next layer in the network. This gating mechanism enables LSTM to selectively remember and process relevant information over longer sequences.

By combining the cell state and gating mechanisms, LSTM effectively overcomes the vanishing gradient problem that plagues traditional RNN. This allows them to learn complex relationships between elements even if they are separated by significant distances within a sequence. This makes LSTM highly effective for various tasks that involve sequential data.

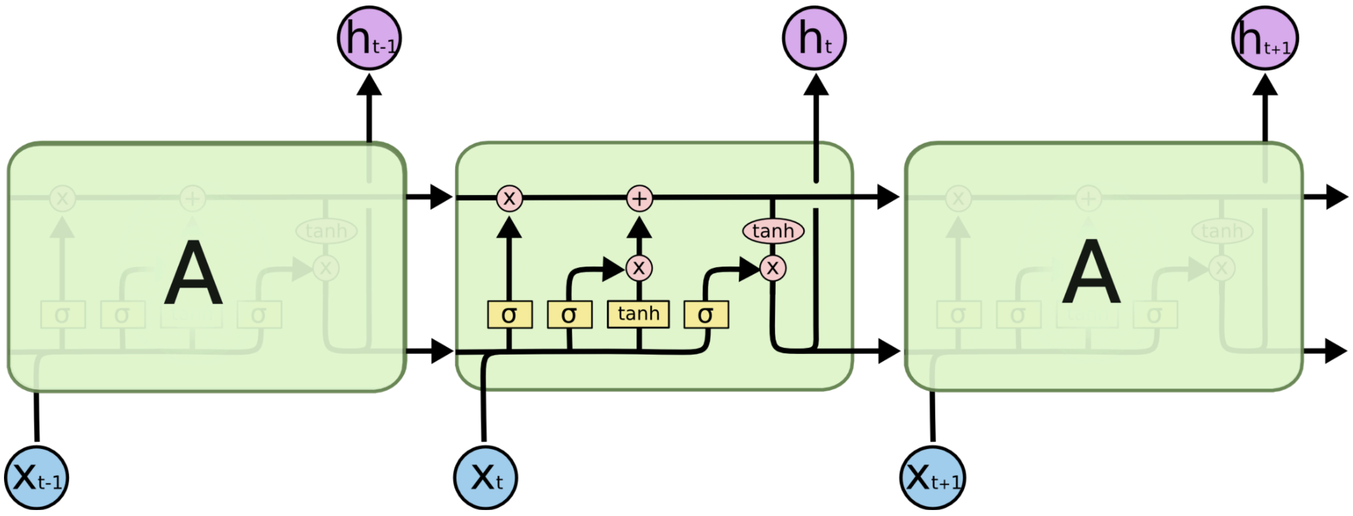
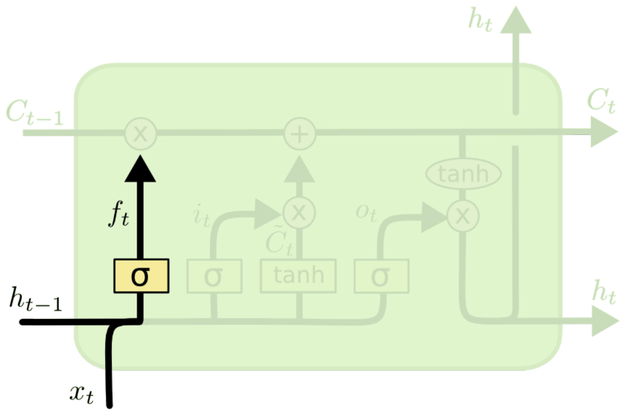


**Figure 8.** LSTM Cell, image from "https://colah.github.io/posts/2015-08-Understanding-LSTMs/"

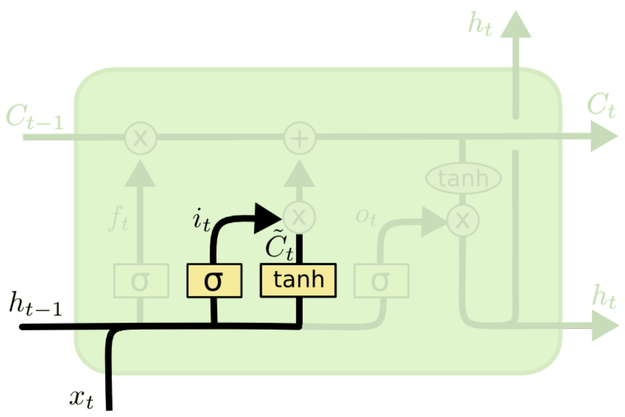
The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer". It looks at $h_{t-1}$ and $x_t$, and outputs a number between 0 and 1 for each number in the cell state $c_{t-1}$. A 1 represents "completely keep this" while a 0 represents "completely get rid of this".



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

**Figure 9.** Forget Gate, image from "https://colah.github.io/posts/2015-08-Understanding-LSTMs/"

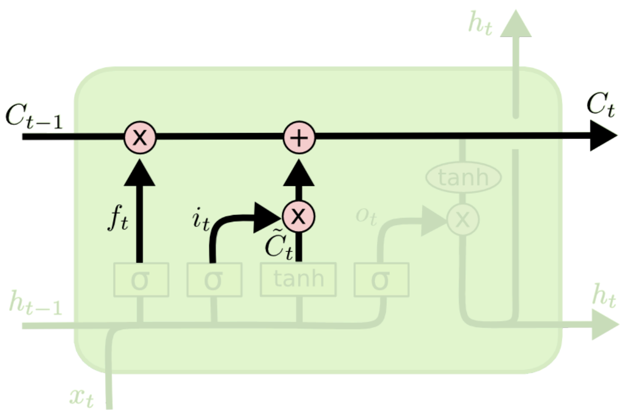
The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, $\overline{C}_t$, that could be added to the state. In the next step, we'll combine these two to create an update to the state.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

**Figure 10.** Input Gate, image from
"https://colah.github.io/posts/2015-08-Understanding-LSTMs/"

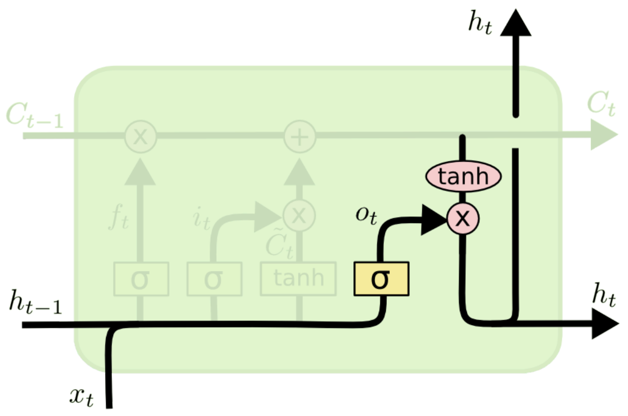
It's now time to update the old cell state,$c_{t-1}$ , into the new cell state $c_t$. The previous steps already decided what to do, we just need to do it. Multiplying the old state by $f_t$, forgetting the things we decided to forget earlier. Then we add $i_t * \overline{C}_t$. These are the new candidate values, scaled by how much we decided to update each state value.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

**Figure 11.** Cell State, image from
"https://colah.github.io/posts/2015-08-Understanding-LSTMs/"

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.



$$o_t = \sigma\left(W_o\,[h_{t-1}, x_t] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

**Figure 12.** Output, image from "https://colah.github.io/posts/2015-08-Understanding-LSTMs/"

## 4.2.3. Bidirectional Long Short-Term Memory:

Bidirectional Long Short-Term Memory (BiLSTM) is a term used for a sequence model that contains two LSTM layers, one for processing input in the forward direction and the other for processing in the backward direction. BiLSTM effectively increases the amount of information available to the network, improving the context available to the algorithm (e.g. knowing what words immediately follow and precede a word in a sentence). It is usually used in NLP-related tasks. The intuition behind this approach is that by processing data in both directions, the model is able to better understand the relationship between sequences.

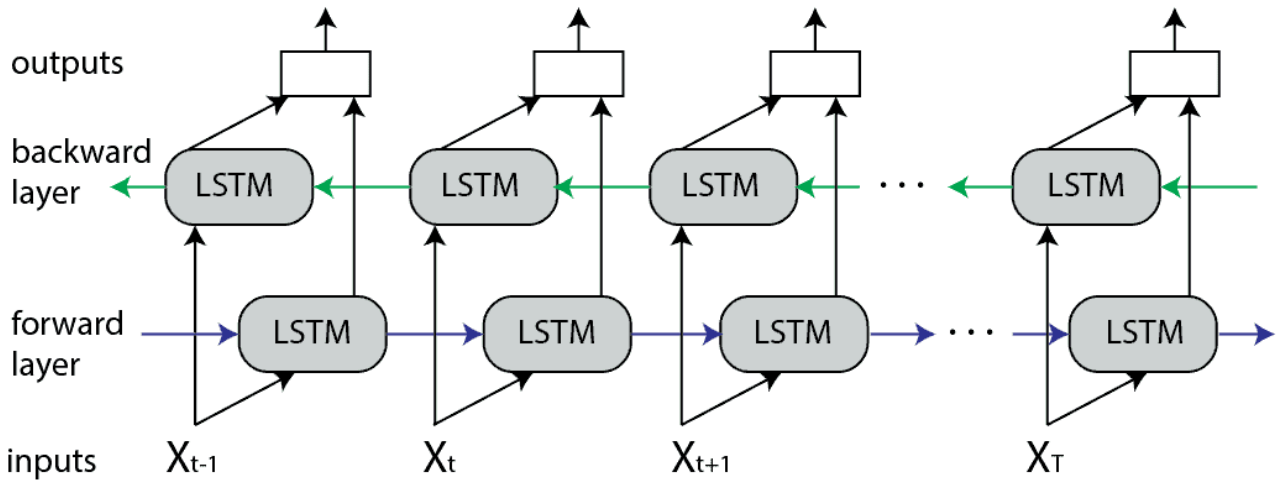


**Figure 13.** Bidirectional Long Short-Term Memory, image from "https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm"

The architecture of bidirectional LSTM comprises two unidirectional LSTM which process the sequence in both forward and backward directions. This architecture can be interpreted as having two separate LSTM networks, one gets the sequence of tokens as it is while the other gets in the reverse order. Both of these LSTM networks return a probability vector as output and the final output is the combination of both of these probabilities. It can be represented as:

$$p_t = p_t^f + p_t^b$$

where:

- $p_t$: Final probability vector of the network
- $p_t^f$: Forward probability vector of the network
- $p_t^b$: Backward probability vector of the network

In essence, BiLSTM, by processing information bidirectionally, unlocked a deeper understanding of context within sequences, leading to superior performance in various sequential tasks.

### 4.2.4. Attention Mechanism:

The introduction of Attention Mechanism in deep learning has brought significant efficiency to many models; it has been and continues to be an indispensable component in the most advanced models.
The Attention mechanism, introduced by Bahdanau et al. in 2014, addressed the weaknesses of the use of a fixed-length encoding vector, which limited the decoder's ability to access information within the input. This problem becomes particularly severe for long and/or complex sequences, where the information gets compressed into the same size representation as shorter or simpler ones, potentially losing important details.

The attention network was designed to identify the highest correlations among words within a sentence. This correlation is captured in neuronal weights through backpropagation, either from self-supervised pre-training or supervised fine-tuning. Attention network allows the calculation of the hidden representation of a token equal access to any part of a sentence directly, rather than only through the previous hidden state.

The Attention mechanism of "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE" [20] is illustrated in **Figure 14** below.

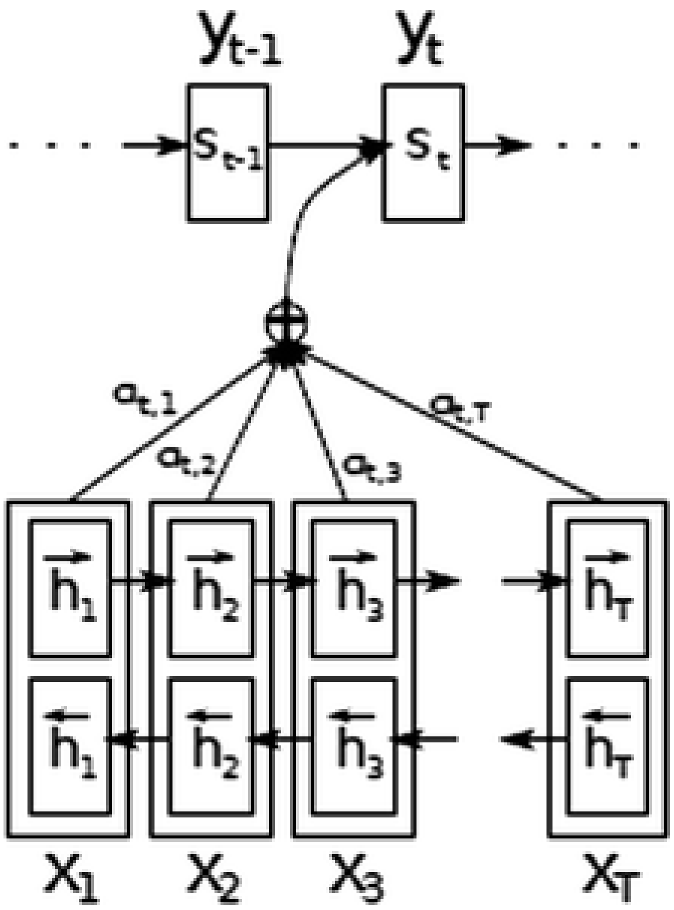


**Figure 14.** Attention Mechanism, image from [20]

**Alignment scores**: The alignment model takes the encoded hidden states, $h_i$, and the previous decoder output, $s_{1-t}$, to compute a score, $e_{t,i}$, that indicates how well the elements of the input sequence align with the current output at the position, t. The alignment model is represented by a function, a(.), which can be implemented by a feedforward neural network:

$$e_{t,i} = a(s_{1-t}, h_i)$$

**Weights**: The weights, $a_{t,i}$, are computed by applying a softmax operation to the previously computed alignment scores:

$$a_t = softmax(e_{t,i})$$

**Context vector**: A unique context vector, $c_i$, is fed into the decoder at each time step. It is computed by a weighted sum of all, T, encoder hidden states:

$$c_i = \sum_{j=1}^{T} a_{t,i} h_i$$

## 4.2.5. Modified Attention Mechanism:

Based on Bahdanau et al. Attention mechanism [20], our team did several modifications as follows:
**Alignment scores**: Same as Bahdanau et al., the alignment model is represented by a function, a(.), which can be implemented by a feedforward neural network. Add a bias $b_i$ factor and put it all in a tanh function:

$$e_{t,i} = tanh(a(s_{1-t}, h_i) + b_i)$$

**Weights:** The weights, $a_{t,i}$, are computed by applying a softmax operation to the previously computed alignment scores:

$$a_t = softmax(e_{t,i})$$

**Context vector:** A unique context vector, $c_i$, is fed into the decoder at each time step. It is computed by multiplying the weight $a_{t,i}$ with encoder hidden states:

$$c_i = a_{t,i} h_i$$

## 4.2.6. Multihead Attention:

Multihead Attention, introduced in [23], is a module for attention mechanisms that runs through an attention mechanism multiple times in parallel. Each of these is called an Attention Head. This particular attention is called "Scaled Dot-Product Attention"[23].

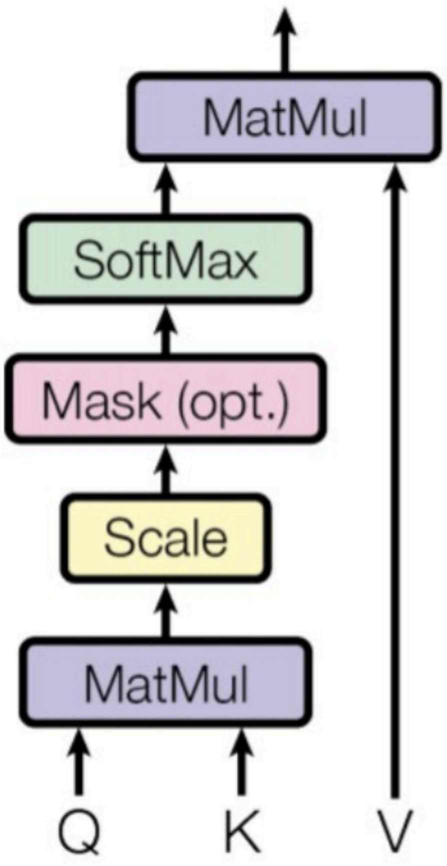


**Figure 15.** Scaled Dot-Product Attention, image from [23]

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where $d_k$ is the dimension of queries (Q) and keys (K).

To make the Scaled Dot-Product Attention "Multihead", the queries, keys, and values are linearly projected **h** times with different, learned linear projections to $d_k$, $d_k$ and $d_v$ dimensions, respectively. Next, perform the attention function in parallel on each of these projected queries, keys and values, yielding $d_v$- dimensional output values. These independent attention outputs are then concatenated and once again projected, resulting in the final values.

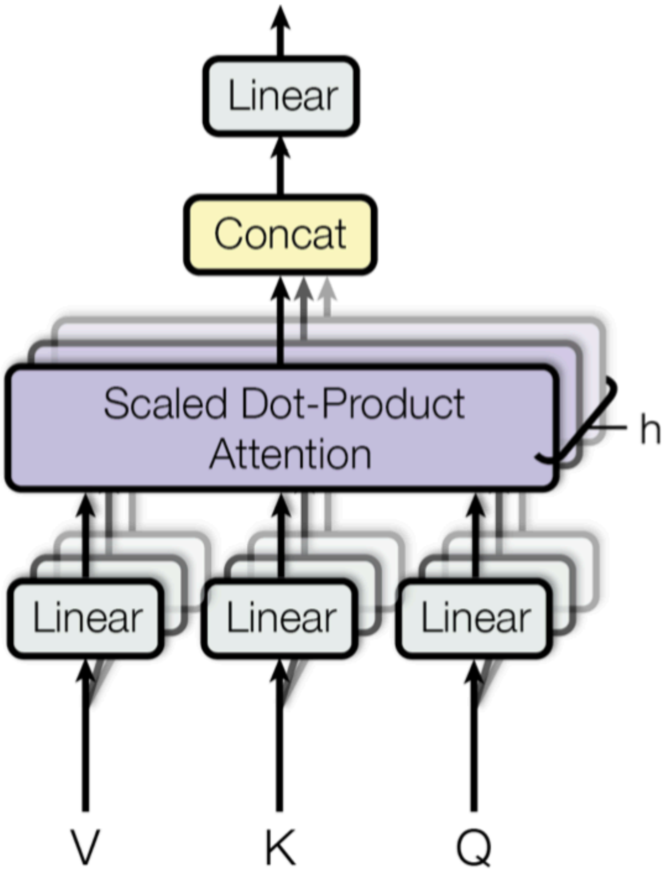


**Figure 16.** Multihead Attention, image from [23]

$$MultiheadAttention(Q, K, V) = Concat(head_1. head_2,..., head_k)W^O$$

where $head_i = Attention(QW_i^Q, KW_i^k, VW_i^V)$

Where the projections are parameter matrices $W_i^Q \in R^{d_{model} \times d_k}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^Q \in R^{d_{model} \times d_v}$ and $W^O \in R^{d_v \times d_{model}}$.

## 4.2.7. Modified Multihead Attention:

Based on Multihead Attention mechanism [23], our team did several modifications as follows:

Instead of Scaled Dot-Product Attention, we using our modified attention mechanism explained in **section 4.2.5**:

$$e_{t,i} = tanh(a(s_{1-t}, h_i) + b_i)$$

Inorder to make the Attention "Multihead", $h$ attention head and then projected. Next, instead of concatenating the "heads" like in the normal Multihead Attention, a summation is performed:

$$MultiheadAttention(e_{t,i}) = Sum(head_1 . head_2, ..., head_k)W^O$$

Add bias and put the whole process in a tanh function:

$$MultiheadAttention(e_{t,i}) = tanh(Sum(head_1, ..., head_k)W^O + b_i^O)$$

The weights, $a_t$, are computed by applying a softmax operation to the previously computed alignment scores:

$$a_t = Softmax(MultiheadAttention(e_{t,i}))$$

Finally, the context vector $c_i$ is calculated by by multiplying the weight $a_{t,i}$ with the encoder hidden states $h_i$:

$$c_i = a_{t,i} h_i$$

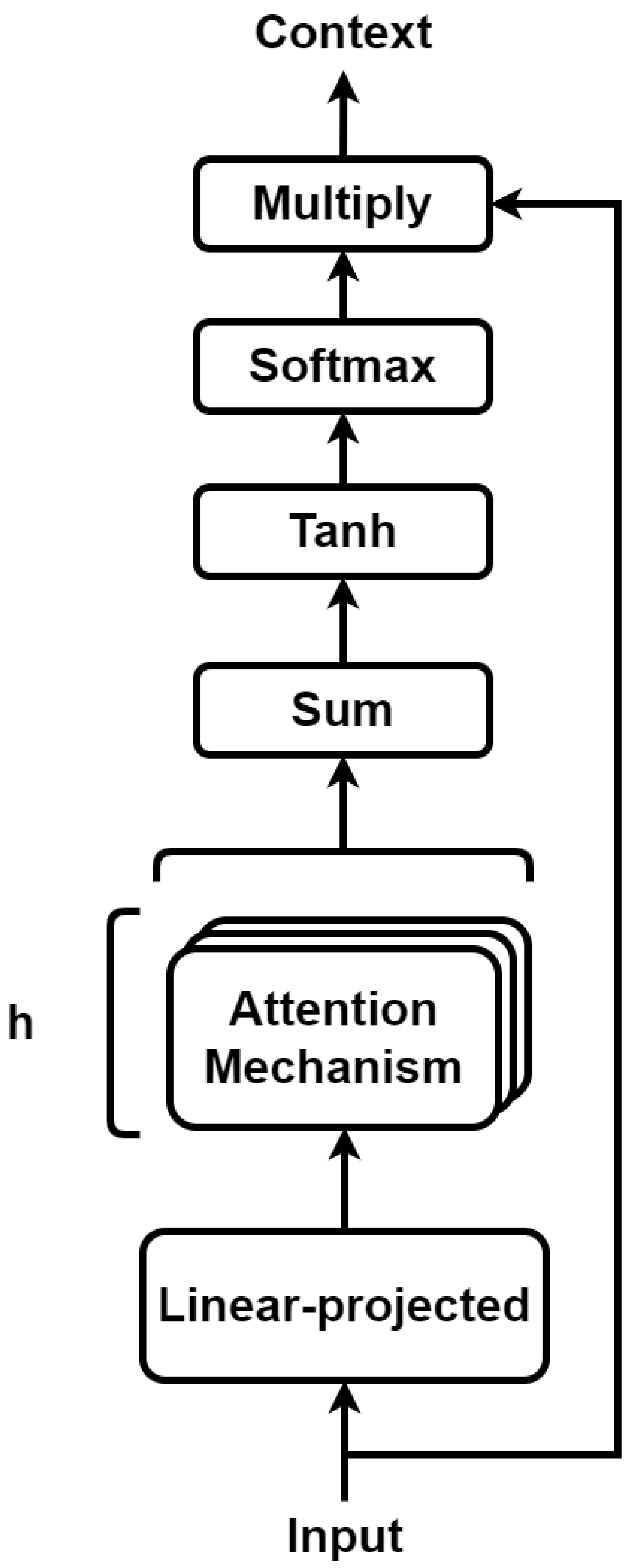


**Figure 17.** Modified Multihead Attention

## 4.2.8. Tanh Activation:

A tanh activation function is a hyperbolic tangent activation function. It transforms the continuous real input values into a range of (-1,1). Tanh function is defined by:

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

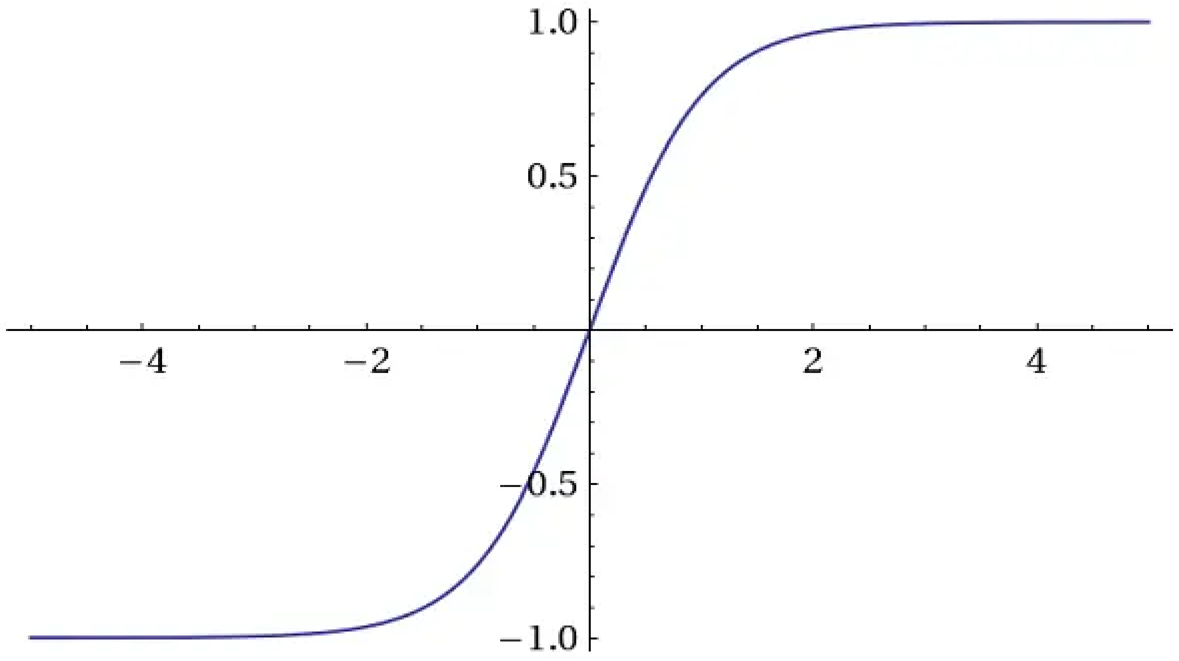


**Figure 18.** Tanh Function, image from "https://vidyasheela.com/post/hyperbolic-tangent-tanh-activation-function-with-python-code"

## 4.2.9. Sigmoid Activation:

A sigmoid function is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point. It transforms the continuous real input values into a range of (0,1). Sigmoid functions are defined by:

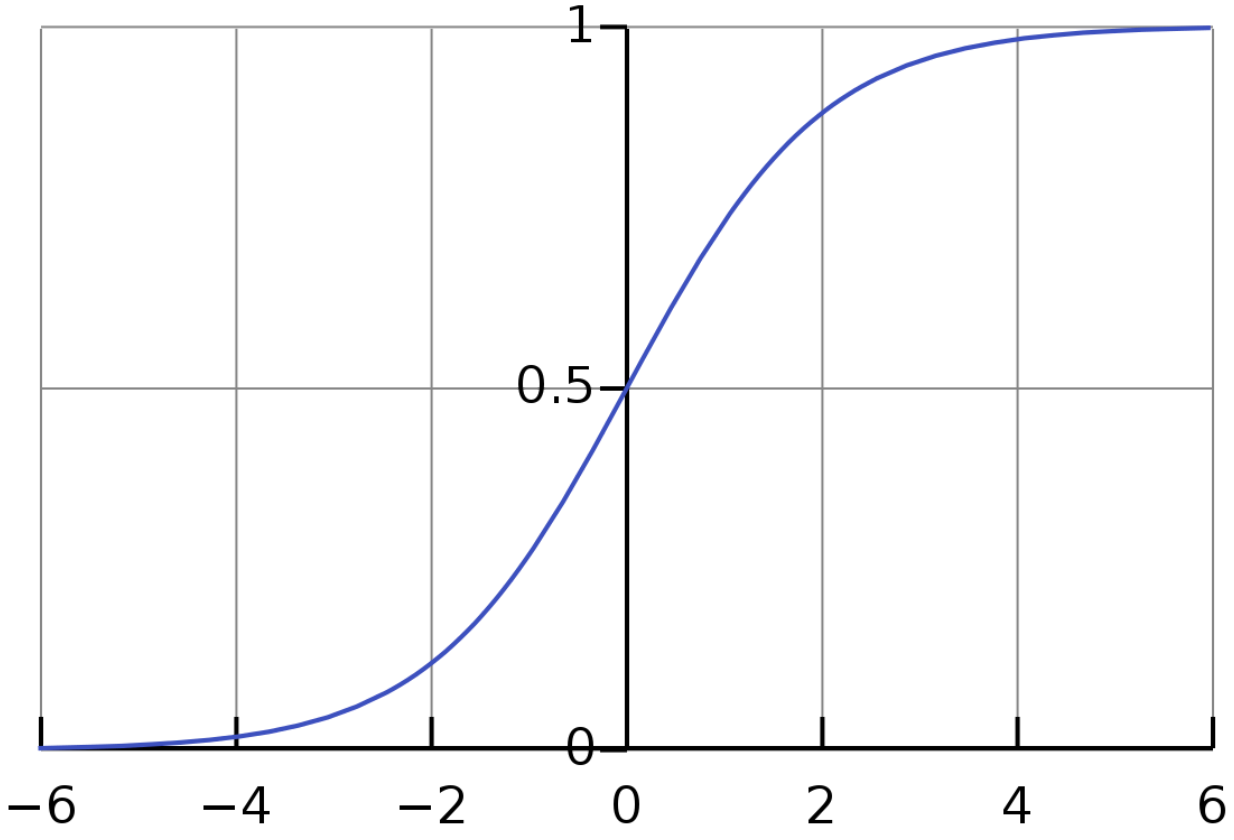


**Figure 19.** Sigmoid Function, image from "https://en.wikipedia.org/wiki/Sigmoid_function"

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

## 4.2.10. Binary Cross-Entropy:

Binary Cross Entropy (BCE) is a loss function that's normally used for Binary classification problems. These problems answer questions with only 2 choices (yes or no, A or B, 0 or 1, left or right) for example the problem of classifying dogs and cats or classifying people and horses. While BCE is ideal for these simple choices, it can also be applied to situations where an image can have multiple labels at the same time.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

where $y_i$ is the label and $p(y_i)$ is the predicted probability of class 1 for all N points.

### 4.2.11. Weighted Binary Cross-Entropy:

Weighted Binary Cross-Entropy (WBCE) applies a scaling parameter to Binary Cross Entropy, allowing us to penalize false positives or false negatives more harshly. It is also effective in handling imbalanced datasets by assigning higher penalties for misclassifying the minority class. WBCE is defined by:

$$WBCE = -E[w_1 \cdot y_i \cdot log(p(y_i)) + w_0 \cdot (1 - y_i) \cdot log(1 - p(y_i))]$$

where $y_i$ is the label and $p(y_i)$ is the predicted probability of class 1 for all N points, $w_1$ and $w_0$ is the weight parameters of the class 1 and 0 respectively.

### 4.2.12. Early Stopping:

After every epoch, the validation set was evaluated in order to stop the training if the evaluation of valid loss did not improve for more than ten (10) epochs, avoiding possible over-fitting, as the validation set was composed of unseen songs.

### 4.2.13. F1-score:

The metrics used to evaluate the accuracy of the Automatic Music Transcription model is the Precision, Recall and F1-score. Precision shows how often an ML model is correct when predicting the target class. Recall shows whether an ML model can find all objects of the target class. F1-score shows the harmonic mean of the Precision and Recall of a classification ML model. The method to calculate those metrics are shown as below:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

where TP, FP, FN stand for True Positive - number of positive samples which are correctly predicted (over all samples), False Positive - number of negative samples incorrectly predicted, and False Negative - number of positive samples incorrectly predicted, respectively. This measure is bounded by 0 and 1, with 1 corresponding to the correct transcription.

### 4.2.14. Post-processing

Post-processing could be done using complex but really effective techniques such as using an adaptive threshold during training to maximize the accuracy (instead of simply rounding the predictions) or training HMM models. However, in order to adapt to the time and job allocation constraints of this project, a simple post-processing was designed. The main idea was to erase the predicted pitches which duration was too short to be considered as a pitch and fill small gaps between pitches.

All the mentioned methods will be used for the experiment to find the best method to suit the purpose of this project, except Attention (**section 4.2.4**) and Multihead Attention (**section 4.2.6**) are the base methods to be modified, the post-processing method (4.2.14) is not completed, it will not take account in calculating accuracy and will only be used when predicting.

# 5. EXPERIMENTS

## 5.1. Base model Long Short-Term Memory

In [17], two different deep learning model structures DNN and LSTM are compared. DNN and LSTM were used, with 1,2,3, and 4 hidden layers respectively. All of them had 256 units in each hidden layer. The project [17] has proved the performance of DNN over LSTM and the suitable number of hidden layers is 3. Unsatisfied with the result, LSTM is our base model.

However, originally, in [17],  Mean Squared Error (MSE) between the output and the labels vector for each frame was set as the loss function, all the hidden layers used tanh activation and the output layer's activation was set as ReLU as both the target output.

Through research, our team first determined that using ReLU activation in the last output layer and MSE as a loss function is not the optimal choice for the current Multilabel-Classification tasks. Given the limitations of ReLU for multi-label classification, we opted to employ a Sigmoid activation function in the final output layer. Sigmoid functions are well-suited for this task because they output values between 0 and 1, which can be interpreted as probabilities for each class.

MSE is used to check how close estimates or forecasts are to actual values. The lower the MSE, the closer it forecasts to the actual. Since MSE is primarily concerned with minimizing the squared difference between a single predicted value and the actual value, in multi-label classification, however, multiple labels can be true simultaneously. MSE may struggle to capture the overall error associated with each individual class. As a consequence, Binary Cross-Entropy (BCE) takes place. However, in this case, we used the Weighted Binary Cross-Entropy (WBCE) described in **section 4.2.11**.

Our modification for the model was to replace the ReLU activation layer with Sigmoid and MSE with BCE. Each of the LSTM layers had 256 hidden units and tanh activation function. Dropout rate was set to 20% (0.2), early stopping was done using the accuracy on the validation set in order to avoid possible over-fitting. Adam optimizer was used with a default learning rate of 0.001. Loss function is WBCE described in **section 4.2.11**, with 1 to 0 ratio 0.65, 0.35.

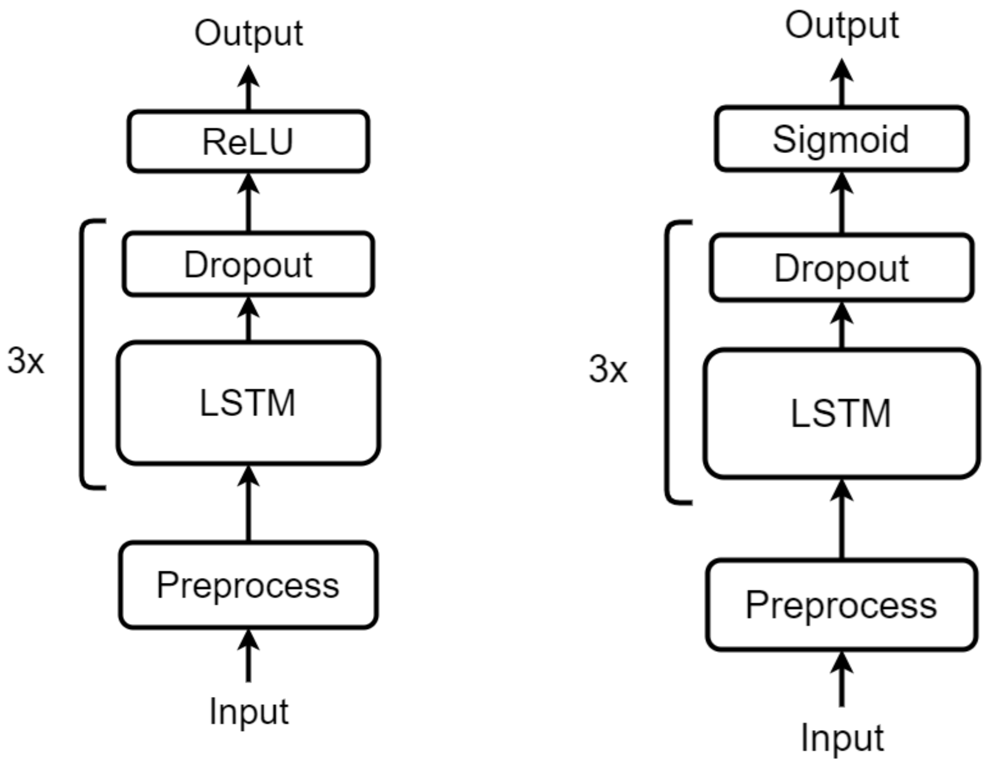


**Figure 20.** Base Model & Modified Base Model

## 5.2. Bidirectional Long Short-Term Memory

While Long Short-Term Memory (LSTM) networks excel at capturing sequential information, Bidirectional LSTM (BiLSTM) offers a distinct advantage in sequential tasks. Different from it's unidirectional counterparts - LSTM, which analyzes sequences in one direction (typically forward), BiLSTM processes the music information in both forward and backward directions. This capability allows BiLSTM to leverage contextual information from both the preceding and succeeding notes within a musical sequence. This is particularly beneficial in AMT because musical meaning often depends on the interplay between notes in both directions. For example, a resolving chord at the end of a phrase can influence the interpretation of earlier notes in the sequence. By considering the entire musical context, BiLSTM can achieve superior performance in tasks like pitch and onset detection, chord recognition, and overall music transcription accuracy compared to traditional LSTM.Aware of its characteristics, our team conducted an experiment to test the performance of BiLSTM and compare it with normal unidirectional LSTM. With the same model architecture as normal LSTM: 3 BiLSTM hidden layers along with 3 Dropout layers after each BiLSTM layer, 1 Sigmoid activation function in the final output layer. The model structure is described in **Figure 21** below.

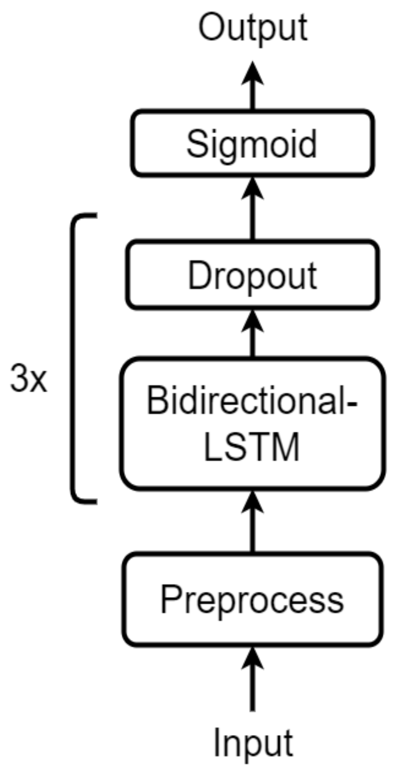


**Figure 21.** Bidirectional Long Short-Term Memory Model

All of the LSTM layers had 256 units in each hidden layer, which means there are 512 units in each BiLSTM layer with tanh activation function. Dropout rate was set to 20% (0.2), early stopping was done using the accuracy on the validation set in order to avoid possible over-fitting. Adam optimizer was used with a default learning rate of 0.001. Loss function is WBCE described in **section 4.2.11**, with 1 to 0 ratio 0.65, 0.35.

## 5.3. Long Short-Term Memory - Attention

Attention mechanisms enhance deep learning models by selectively focusing on important input elements, improving prediction accuracy and computational efficiency. They prioritize and emphasize relevant information, acting as a spotlight to enhance overall model performance. Moreover, Attention-based architectures currently lay at the basis of state-of-the-art NLP models. As a consequence, our team decided to run an experiment on applying Attention mechanism to our model.

Upon testing with the General Attention (Scaled Dot-Product Attention) [23], our team recognised that such Attention's mechanism is not suitable for our model. Through research,

our team has applied and modified the Attention mechanism of [20] to best suit our current model. The modified Attention mechanism that we are using is explained in **section 5.5**.
Also inspired by the paper [20], which stacked Attention between 2 Recurrent Neural Networks, by replacing RNN with LSTM, our model structure is illustrated in **Figure 22**.

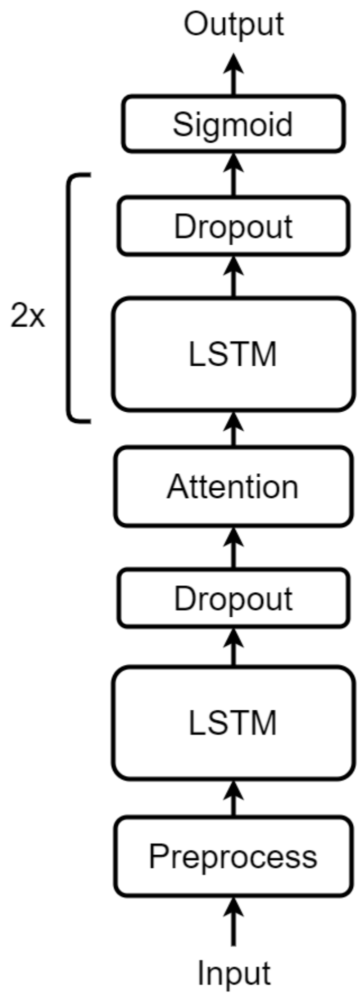


**Figure 22.** Long Short-Term Memory - Attention Model

All of the LSTM layers had 256 units in each hidden layer with tanh activation function. Attention mechanism was explained in **section 4.2.5**. Dropout rate was set to 20% (0.2), and early stopping was done using the accuracy on the validation set in order to avoid possible over-fitting. Adam optimizer was used with a default learning rate of 0.001. Loss function is WBCE described in **section 4.2.11**, with 1 to 0 ratio 0.65, 0.35.

## 5.4. Bidirectional Long Short-Term Memory - Attention

Just like in **section 5.2**, by replacing the LSTM layers with BiLSTM, we conducted an experiment to test the performance of BiLSTM as well as to see whether our Attention mechanism interacts with Bidirectional objects. The model structure is illustrated in **Figure 23** below.

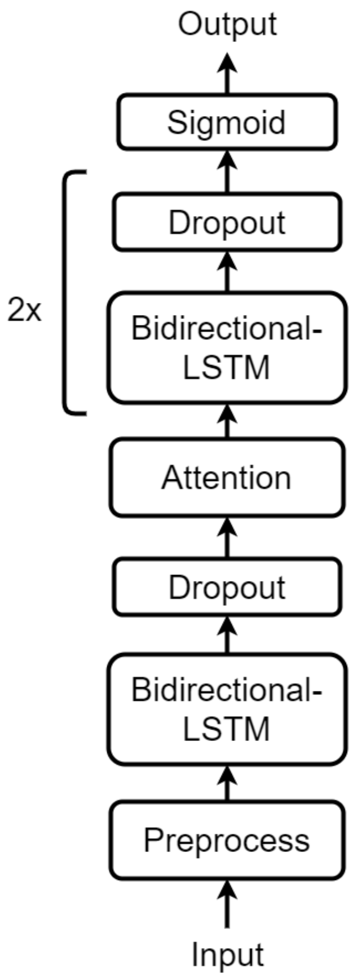


**Figure 23.** Bidirectional Long Short-Term Memory - Attention Model

All of the LSTM layer had 256 units in each hidden layer, which means there are 512 units in each BiLSTM layer with tanh activation function. Attention mechanism was explained in 4.2.5. Dropout rate was set to 20% (0.2), early stopping was done using the accuracy on the validation set in order to avoid possible over-fitting. Adam optimizer was used with a default learning rate of 0.001. Loss function is WBCE described in **section 4.2.11**, with 1 to 0 ratio 0.65, 0.35.

## 5.5. Attention - Long Short-Term Memory

A Transformer is a model architecture that eschews recurrence and instead leverages an attention mechanism to capture global dependencies within the input sequence. While classic sequence-to-sequence models often utilize encoders and decoders built with RNN or CNN, the Transformer retains this encoder-decoder structure but replaces recurrent connections with attention mechanisms, enabling significant efficiency gains. This approach allows for significantly more parallelization during training compared to recurrent or convolutional methods.

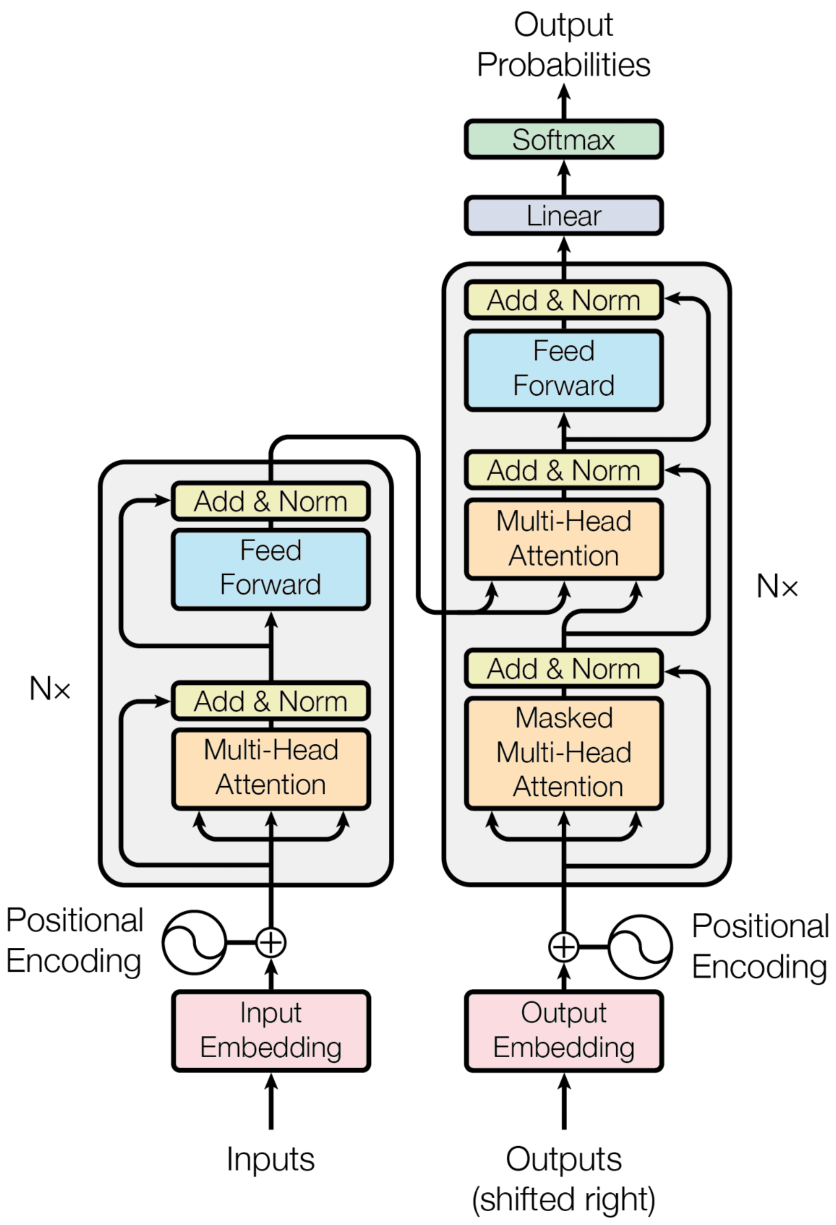


**Figure 24.** Transformer architecture, image from [23]

Throughout the last few years, the interest in Transformers has grown considerably and so does the number of papers focused on its possible application. The Transformer model has revolutionized the field of natural language processing (NLP) by introducing a novel architecture that relies solely on attention mechanisms. This approach has demonstrated exceptional performance in various tasks, including machine translation and text summarization. Motivated by the success of the Transformer architecture in various sequence processing tasks, we modified our previous LSTM-Attention-LSTM model. The key modification is switching the Attention layer to the beginning of the network. This placement allows the model to attend to critical information earlier in the processing sequence, potentially leading to enhanced performance. Moreover, a DNN layer was added before the Sigmoid function as projected to the Linear layer of the Transformer. The model's structure is illustrated in **Figure 25** below.

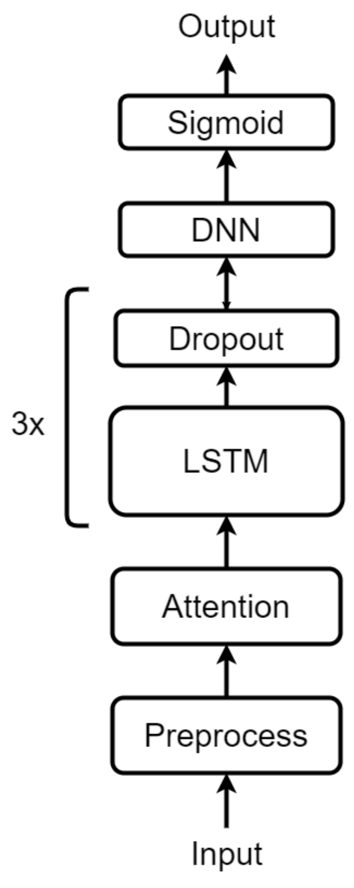


**Figure 25.** Attention - Long Short-Term Memory Model

All of the LSTM layers had 256 units in each hidden layer. Tanh activation function was used in the LSTM and DNN layers. Attention mechanism was explained in **section 4.2.5**. Dropout rate was set to 20% (0.2), early stopping was done using the accuracy on the validation set in order to avoid possible over-fitting. Adam optimizer was used with a default learning rate of 0.001. Loss function is WBCE described in **section 4.2.11**, with 1 to 0 ratio 0.65, 0.35.

## 5.6. Attention - Bidirectional Long Short-Term Memory

After the success of the previous Transformer-inspired Attention-LSTM model in experiment 5.5, as a consequence in experiment 5.4, our team conducted a test on clarifying the performance of BiLSTM over LSTM. With the same model's structure, by replacing the 3 LSTM layers with BiLSTM, our team expected the Attention - BiLSTM model to outperform the LSTM's version. The model's structure is illustrated in **Figure 26** below.

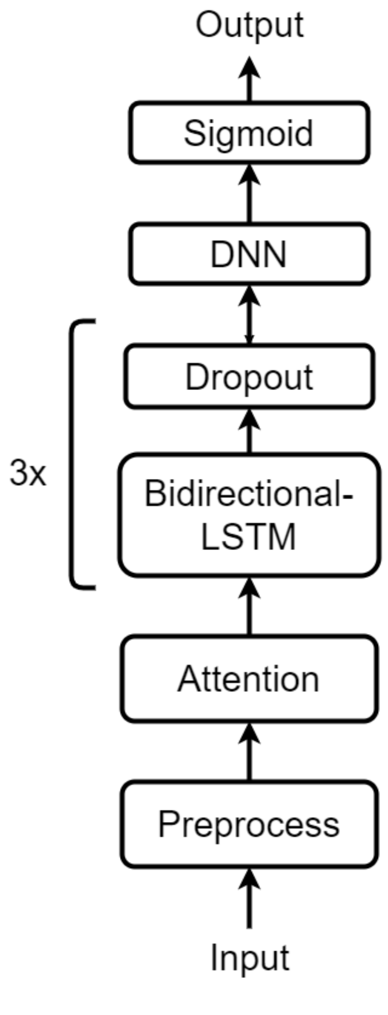


**Figure 26.** Attention - Bidirectional Long Short-Term Memory Model

All of the LSTM layers had 256 units in each hidden layer, which means there are 512 units in each BiLSTM layer. Tanh activation function was used in the LSTM and DNN layers. Attention mechanism was explained in **section 4.2.5**. Dropout rate was set to 20% (0.2), early stopping was done using the accuracy on the validation set in order to avoid possible over-fitting. Adam optimizer was used with a default learning rate of 0.001. Loss function is WBCE described in **section 4.2.11**, with 1 to 0 ratio 0.65, 0.35.

## 5.7. Multihead Attention - Bidirectional Long Short-Term Memory

In Transformer architecture, there are 3 Attention blocks. Multiple attempts were made to add several Attention to the model. However, it seems like this approach is not suitable for our current model. Our team decided to innovate our modified Attention mechanism to make it "Multihead". The configuration is described in **section 4.2.7**. Model's structure is illustrated in **Figure 27** below.

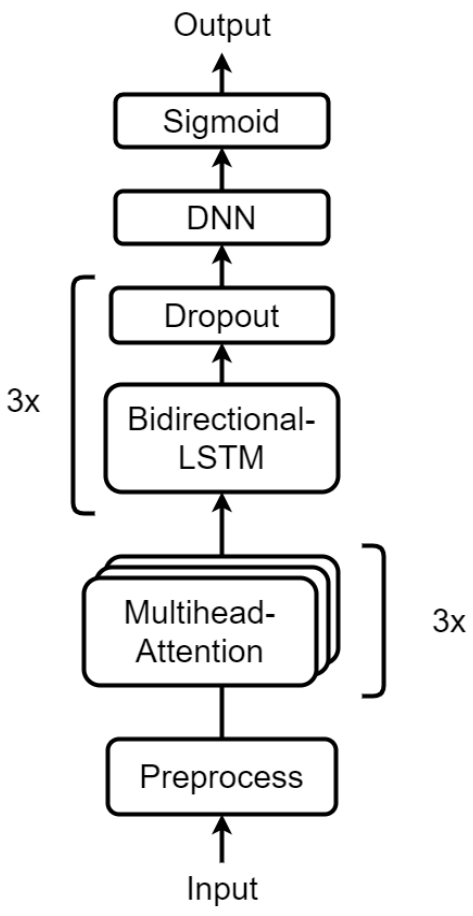


**Figure 27.** Multihead Attention - Bidirectional Long Short-Term Memory Model

All of the LSTM layers had 256 units in each hidden layer, which means there are 512 units in each BiLSTM layer. Tanh activation function was used in the LSTM and DNN layers. Multihead Attention mechanism was explained in **section 4.2.7**, in this model, 3 heads were used. Dropout rate was set to 20% (0.2), early stopping was done using the accuracy on the validation set in order to avoid possible over-fitting. Adam optimizer was used with a default learning rate of 0.001. Loss function is WBCE described in **section 4.2.11**, with 1 to 0 ratio 0.65, 0.35.

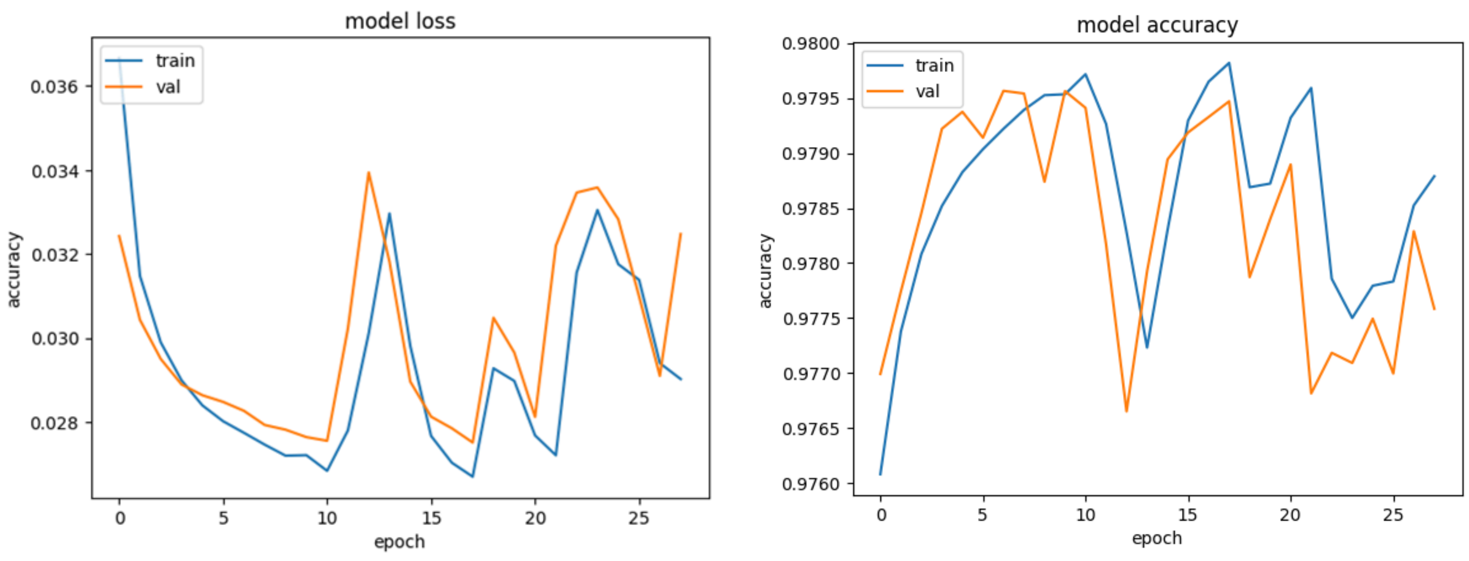
# 6. RESULTS

**Table 5.** Result Table

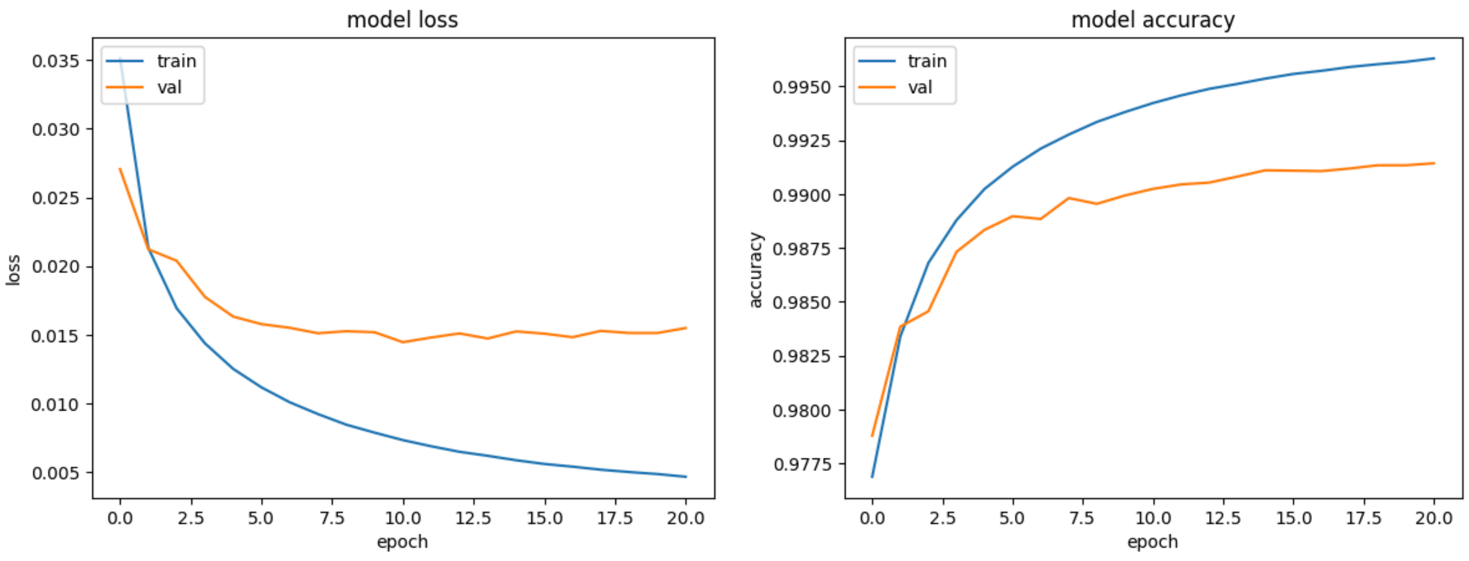
| Model | Parameters | Precision | Recall | Accuracy | F1-score |
|---|---|---|---|---|---|
| LSTM (**Base Model**) | 1.59M | 0.76 | 0.66 | 55.2 | 71.2 |
| Bidirectional LSTM | 4.24M | 0.77 | 0.74 | 61.2 | 75.98 |
| LSTM - Attention | 1.66M | 0.74 | 0.75 | 59.8 | 74.8 |
| Bidirectional LSTM - Attention | 4.35M | 0.8 | 0.83 | 69.5 | 82 |
| Attention - LSTM | 1.66M | 0.80 | 0.84 | 69.8 | 82.2 |
| Attention - Bidirectional LSTM | 4.35M | 0.84 | 0.88 | 75.5 | 86.1 |
| Multihead - Bidirectional LSTM | 4.43M | 0.84 | **0.9** | 77.3 | 87.2 |

| | | | | | |
|---|---|---|---|---|---|
| Multihead - Bidirectional LSTM + Binary Cross Entropy (**Best Model**) | 4.43M | **0.89** | 0.86 | **78.9** | **88.2** |
| Multihead - Bidirectional LSTM + Single Dataset | 4.43M | 0.83 | 0.87 | 74.5 | 85.4 |

Multihead-Bidirectional LSTM model stands out to be the most efficient model and will be used to demonstrate prediction ability, comparative when the model is used with BCE and WBCE are shown in **section 6.1**, when model trains with single and multiple dataset are shown in **section 6.2**.

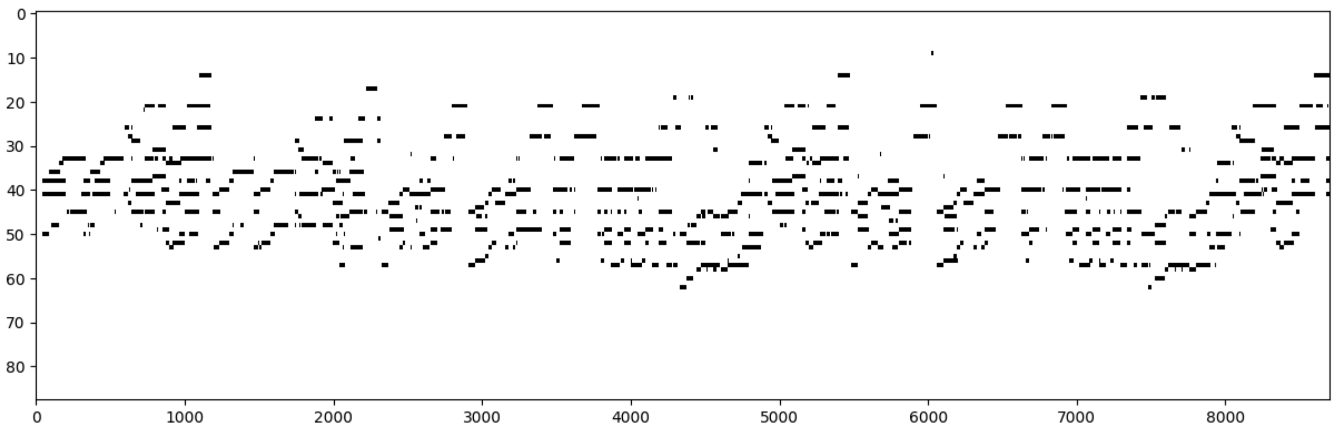


**Base Model Loss and Accuracy in train**

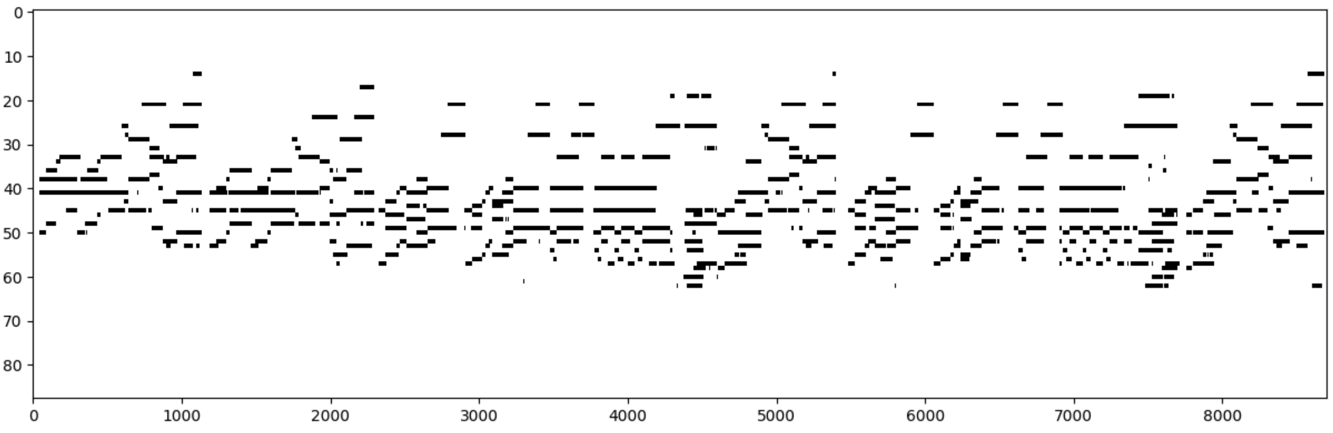


**Best Model Loss and Accuracy in train**

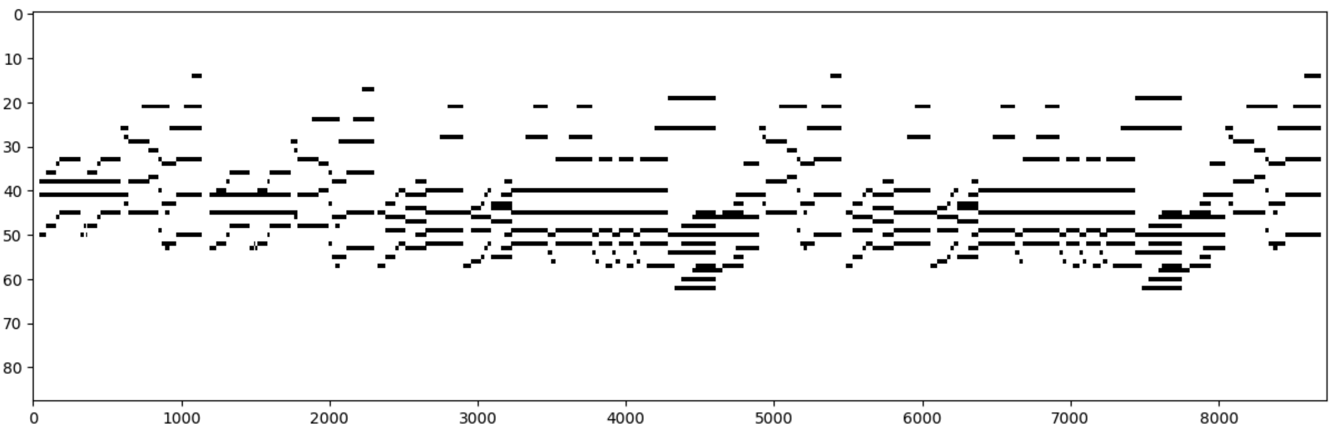
**Figure 28.** Loss and Accuracy in train of Base Model and Best Model



**Predicted result of Base Model**



**Predicted result of Best Model**

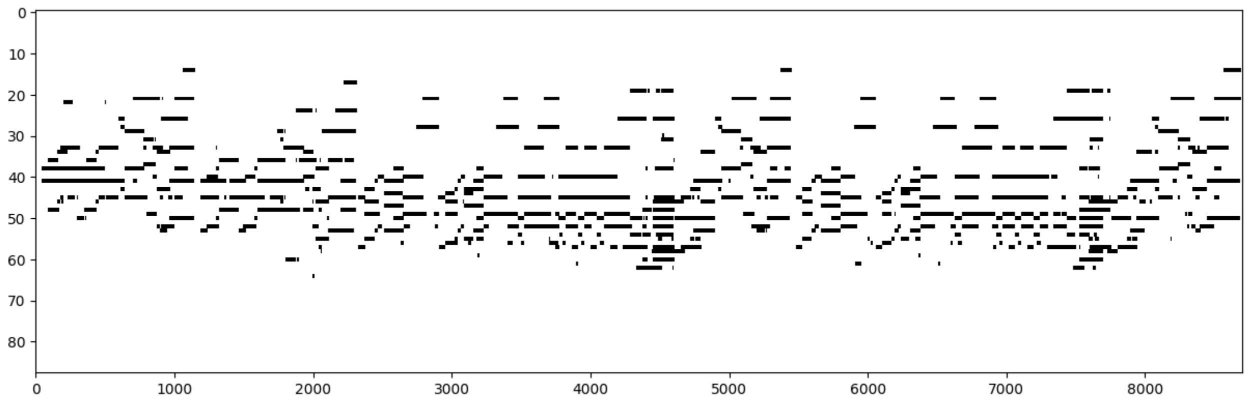


**Ground truth value**

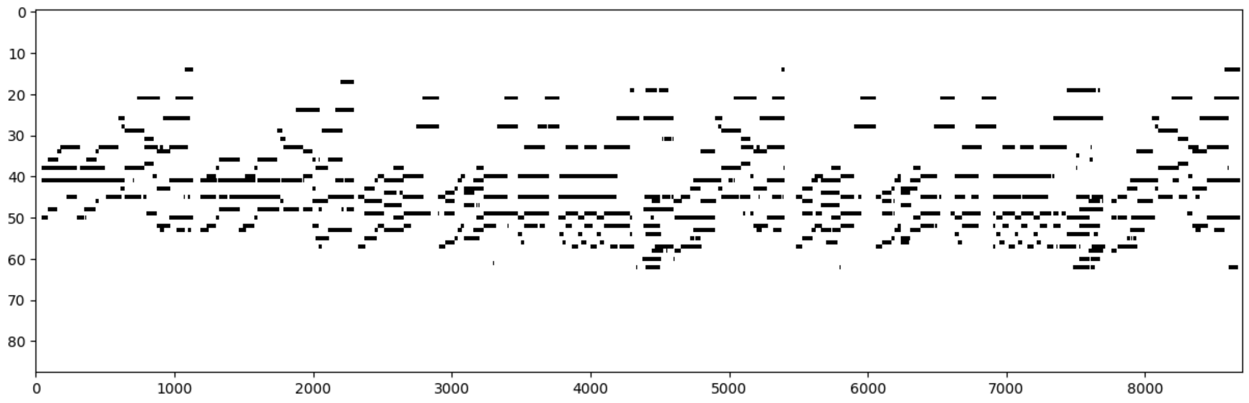
**Figure 29.** Predicted result of Base Model, Best Model vs Ground Truth value

# *EXTRA*

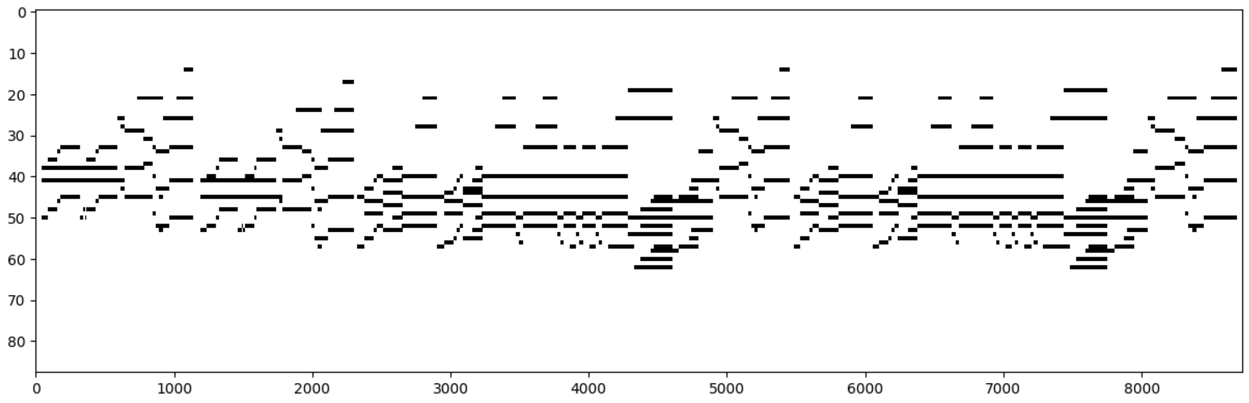
## 6.1. Weighted Binary Cross-entropy



Predicted result of Multihead model when use WBCE



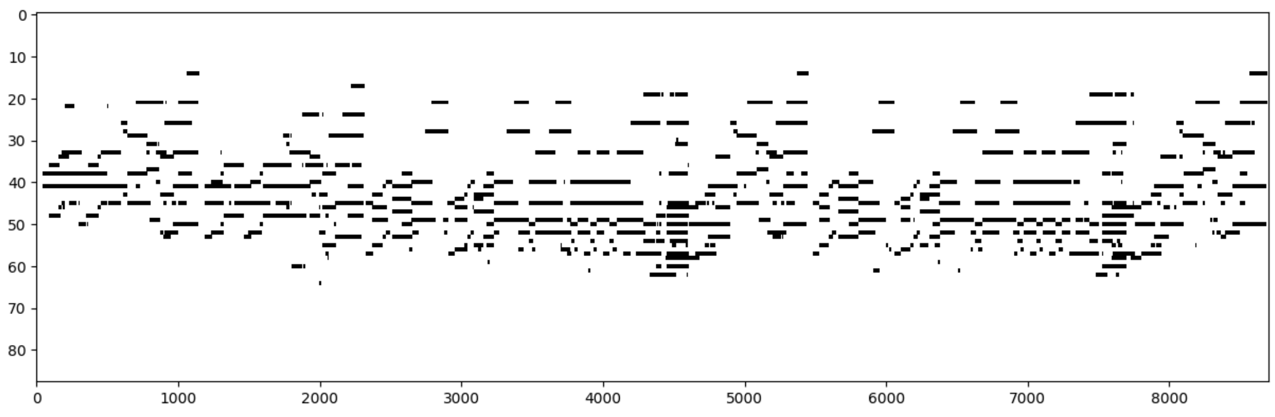
Predicted result of Multihead model when use BCE



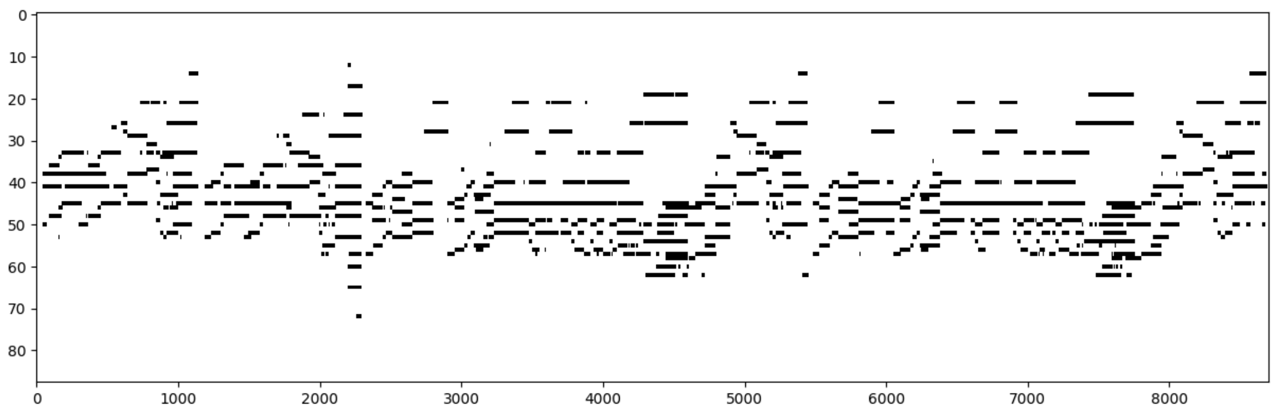
Ground truth value

**Figure 30.** Predicted result of Multihead Attention - BiLSTM when use WBCE and BCE vs Ground Truth value

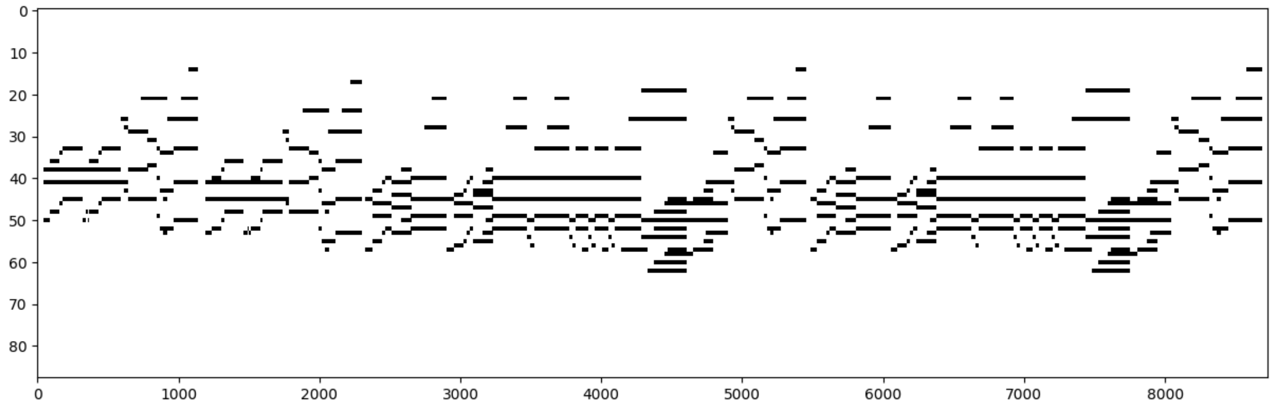
## 6.2. Multihead Attention - Bidirectional Long Short- Term Memory with MAPS only



**Predicted result of Multihead model when use Multiple dataset**



**Prediction Ability of Multihead model when use Single dataset**



**Ground truth value**

**Figure 31.** Predicted result of Multihead Attention - BiLSTM when use Multiple dataset and Single dataset vs Ground Truth value

# 7. DISCUSSION

Our capstone project explored the capability of a deep learning approach for Automatic Music Transcription. By treatting music audio as sequential data, we apply a deep learning method: LSTM for automatic music transcription. In general, the prediction accuracy is promising, given that our neural network is not very deep (~4M parameter) compared to recent approaches. Analyzing its performance, we found that LSTM has an overfitting issue in our tested case. By changing the dropout rates, we can overcome the overfitting issue.

However, due to handling music audio as sequential data, our proposed model adopted several limitations.

## 7.1. Limitation

### 7.1.1. Polyphony

In music, multiple sounds (e.g., instruments, vocals) are played simultaneously (in our case, multiple notes are played at the same time). Unlike in Nature Language Processing (NLP) where words occur one after another. LSTM struggles to capture these concurrent notes relationships effectively.

### 7.1.2. Tremolo

The repetition of a single note, as fast as possible, is called **tremolo**. It is a music performance technique in which a performer plays an individual note or two alternating notes as fast as possible. However, when notes are played very close together in time, their frequencies can overlap in the audio spectrum. This creates difficulty for the AMT system to isolate the fundamental frequency of each note and distinguish between a single sustained note and the rapid tremolo.

### 7.1.3. Up-Tempo

Tempo in music, means how fast the music is played, and Up-Tempo is a piece of music which plays fast or very fast, our model has a low accuracy when predicting such music pieces.

### 7.1.4. Post-processing

As previously mentioned in **section 4.2.14**, our current method of post-processing works by erasing the predicted pitches which were too short and filling small (1-2 frame) gaps between pitches.

## 7.2. <u>Future work</u>

To further increase the accuracy of our music transcription, the neural networks could be trained on a larger dataset, including isolated notes, chords and monophonic pieces of music other than polyphonic pieces of music. Improving post-processing mechanisms could be a promising solution. Moreover, using an adaptive threshold during training to maximize the accuracy instead of simply rounding the prediction. Another way is to create a "deeper" model which indicates more information rather than just which notes are played in a frame.

# 8. CONCLUSIONS

This project explored the potential of deep learning for Automatic Music Transcription (AMT). The Deep Learning model, particularly the Multihead Attention BiLSTM model, achieved promising results in transcribing piano music. Compared to the base-line model LSTM, our models demonstrated improved accuracy in capturing notes and musical structures. Overall, Deep Learning shows a significant promising approach for AMT. By addressing the current challenges and exploring new avenues of research, AMT technology can continue to evolve and offer valuable tools for musicians, composers, music educators, and anyone passionate about music analysis and creation.

**CONFLICTS OF INTEREST:**

All authors declare no conflicts of interest.

# 9. REFERENCES

1. J.A. Moorer. On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer. PhD thesis, On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer, 1975.
2. J.A. Moorer. On the transcription of musical sound by computer. Computer Music Journal, 2(1):7–11, 1977.
3. Masakata Goto and Yoichi Muraoka. A beat tracking system for acoustic signals of music. ACM International Conference on Multimedia, pages 365–372, 1994.
4. Anssi Klapurri, "Introduction to music transcription. Technical report, Institute of Signal Processing, Tampere University of Technology", Korkeakoulunkatu 1, 33720 Tampere, Finland, 2006.
5. V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," IEEE Transactions on Audio, Speech, and Language Processing, vol. 18, no. 6, pp. 1643–1654, 2010.
6. L. Su and Y.-H. Yang, "Combining spectral and temporal representations for multipitch estimation of polyphonic music," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 23, no. 10, pp. 1600– 1612, Oct 2015
7. Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," IEEE Transactions on Audio, Speech, and Language Processing, vol. 18, no. 8, pp. 2121–2133, 2010.
8. P. H. Peeling, A. T. Cemgil, and S. J. Godsill, "Generative spectrogram factorization models for polyphonic piano transcription," IEEE Transactions on Audio, Speech, and Language Processing, vol. 18, no. 3, pp. 519–527, March 2010.
9. P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2003, pp. 177–180.
10. E. Vincent, N. Bertin, and R. Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," IEEE Transactions on Audio, Speech, and Language Processing, vol. 18, no. 3, pp. 528–537, 2010.
11. E. Benetos and S. Dixon, "Multiple-instrument polyphonic music transcription using a temporally-constrained shift-invariant model," Journal of the Acoustical Society of America, vol. 133, no. 3, pp. 1727–1741, March 2013.
12. B. Fuentes, R. Badeau, and G. Richard, "Harmonic adaptive latent component analysis of audio and application to music transcription," IEEE Transactions on Audio, Speech, and Language Processing, vol. 21, no. 9, pp. 1854–1866, Sept 2013.
13. S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 24, no. 5, pp. 927–939, May 2016.
14. R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer, "On the potential of simple framewise approaches to piano transcription," in Proc. International Society for Music Information Retrieval Conference, 2016, pp. 475–481.
15. K. Ullrich and E. van der Wel, "Music transcription with convolutional sequence-to-sequence models," International Society for Music Information Retrieval, 2017.
16. G. E. Poliner and D. P. Ellis, "A discriminative model for polyphonic piano transcription," EURASIP Journal on Applied Signal Processing, vol. 2007, no. 1, pp. 154, 2007.
17. Diego González Morín, "Deep neural networks for piano music transcription,", 2017. Available: https://github.com/diegomorin8/Deep-Neural-Networks-forPiano-Music-Transcription.
18. S. Sigtia, E. Benetos and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," IEEE/ACM Trans. Audio Speech Lang. Process., 24, 927–939, 2016. Available: https://arxiv.org/abs/1508.01774.

19. Luoqi Li, Isabella Ni, Liang Yang. “Music Transcription Using Deep Learning” [ Stanford University], 2017. Available:https://cs229.stanford.edu/proj2017/final-reports/5242716.pdf?fbclid=IwAR2mQj0kGGSnmCUGCIx_rhChWoJgBiZKLtzlUcdMhFnTZecXn4ZFBymz52A
20. D. Bahdanau, K. Cho, and Y. Bengio. “Neural machine translation by jointly learning to align and translate“. In ICLR, 2015.
21. J.C. Brown, “Calculation of a constant Q spec-tral transform,” Journal of the Acoustical Society of America, 1991.
22. Benetos, Emmanouil, et al. "Automatic music transcription: An overview." IEEE Signal Processing Magazine 36.1 (2018): 20-30.
23. Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30, 2017.
24. MAPS Database: A Piano Database for Multipitch Estimation and Automatic Transcription of Music, 2010.
25. The MAESTRO Dataset, 2018
26. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
27. David E. Rumelhart; James L. McClelland, "Learning Internal Representations by Error Propagation" in Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations , MIT Press, 1987, pp.318-362.