國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

使用序列模型將樂譜轉譯為吉他譜

Transcribe Pitch Set Sequence to Guitar Tablature by Sequence to Sequence Model

陳見齊

Jian-Chi Chen

指導教授: 許永真博士 & 項潔博士

Advisor: Jane Yung-jen Hsu, Ph.D. & Jieh Hsiang, Ph.D.

中華民國 113 年 8 月

August, 2024

doi:10.6342/NTU202403031

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

使用序列模型將樂譜轉譯為吉他譜

Transcribe Pitch Set Sequence to Guitar Tablature by Sequence to Sequence Model

本論文係陳見齊君（學號 R10922130）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 113 年 7 月 15 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department of Computer Science and Information Engineering on 15 July 2024 have examined a Master's thesis entitled above presented by CHEN, JIAN-CHI (student ID: R10922130) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

許永真 _____ _____

（指導教授 Advisor）

黃稚靜 楊智淵 _____

_____ _____ _____

系主任/所長 Director: 陳祝嵩

doi:10.6342/NTU202403031

誌謝

在碩士這段過程中，有許多人的幫助與支持值得我深深感謝。首先，我要特別感謝我的指導教授許永真老師，對於我的研究以及論文撰寫都提供了相當重要的方向引導與實質建議。也感謝老師願意指導我這位大學主修化學工程的跨領域學生，對於我想要時間補足我不足的知識與技術基礎也持開放態度。

也感謝我就讀師大附中時認識的 1198 班羅逸翔學長，在我人生迷茫的時候告訴我電腦科學的有趣之處，甚至教我如何撰寫 C++ 程式，如果沒有這段經歷我可能不會踏入這個領域。

也感謝研究室的夥伴們豐富了這段過程，特別是 Erick、政倫、揚昇。Erick 一直都為實驗室盡心盡力，確保了設備的正常運作，並且樂於與我分享技術與新知，也是有他的協助才能順利擔任 TA; 政倫是我大學時代一起沈迷遊戲的損友，但在我準備碩士入學考試時是指引方向的益友 (雖然在我備考時期還是會找我去打遊戲)，但如果沒有他和我討論，我可能沒辦法這麼順利回到臺大讀書; 再來是揚昇，備考時期還是討論考題的網友，沒想到後來成為同一個實驗室的成員，交流課業、一起當 TA 和 Admin、一起打遊戲，還搭了他好幾次順風車。

最後，感謝我的家人，支持我攻讀碩士的父親以及支持並關心我的阿姨們。在此，向所有支持和幫助過我的人表達我最誠摯的謝意。

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

中文摘要

吉他指法譜的編排是一項專業技能，需要投入大量時間學習，甚至需要向專家請教。初學者和不熟悉吉他的音樂創作者可能因此卻步。我們希望通過自動化的指法轉譯系統，降低指法編排的門檻，讓更多人享受吉他演奏的樂趣。

因此，本研究提出了一種將樂譜 (Sheet Music) 作為輸入，吉他指法譜作為輸出的機器學習模型。

由於音樂是一種具有時序且前後相關的資訊，適合使用序列模型 (Sequence Model) 處理，我們設計了基於 Transformer 架構的模型進行轉譯工作。我們採用 dadaGP 資料集 (包含 26181 個不同內容的 GuitarPro 檔案) 作為訓練資料集。我們先過濾多餘的檔案，並透過 PyGuitarPro 套件讀取吉他指法譜資料。基於經驗和觀察，我們選擇忽略節奏資訊，並以此前提設計了音樂資訊和指法資訊的嵌入方法。此外，我們設計了資料後處理方法，檢查輸出的指法譜，若發現某個指法對應的音高不在輸入資訊中，則改為不彈奏，從而改善模型輸出表現，並成功通過 GPU 平行運算實現，降低了後處理的計算時間。

在實驗中，我們首先嘗試訓練出一個能夠收斂的模型，並以此為基礎，尋找表現較好的參數和架構設定。實驗結果在表現上優於近年的研究。

關鍵字：吉他、樂譜、指法譜、轉譯

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

英文摘要

Arranging guitar tablature is a specialized skill that requires a significant amount of time to learn and often involves seeking advice from experts. Beginners and music creators unfamiliar with the guitar might be discouraged by these barriers. We aim to lower the barriers of tablature arrangement through an automated tablature transcription system, enabling more people to enjoy playing the guitar.

Thus, this study proposes a machine learning model that takes sheet music as input and produces guitar tablature as output.

Since music is sequential and context-dependent, it is well-suited for sequence models. We designed a model based on the Transformer architecture to handle the transcription task. We used the dadaGP dataset, which contains 26,181 unique GuitarPro files, as our training dataset. We filtered out redundant files and read the guitar tablature data using the PyGuitarPro library. Based on experience and observations, we ignored rhythmic information and designed our embeddings for musical and tablature information accordingly. Additionally, we developed a post-processing method to check the output tablature. If a note in the tablature does not correspond to a pitch in the input, we mark it as unplayed, thus improving the model's performance. This process was successfully implemented using GPU parallel computing, reducing the computation time for post-processing.

In our experiments, we first trained a convergent model and then used it as a baseline to find better parameter settings and model configurations. The results outperformed recent study in terms of performance.

Keywords: Guitar, Sheet Music, Tablature, Transcription

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

目次

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

圖次

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

表次

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

第一章　緒論

在本章，我們想先介紹背景知識: 指法譜的形式以及指法譜的特性，接著說明我們的研究與動機，最後將簡略地介紹我們想借鏡的方法: 自然語言處理中的文字翻譯任務，以及我們認為可以借鏡此方法的原因。

1.1　背景

在這節中，首先會介紹指法譜的形式，再解釋指法譜的特性。

1.1.1　何謂指法譜 (Tablature)

有別於常見的樂譜 (Sheet Music)，吉他指法譜 (Guitar Tablature)，是一種記錄演奏指法的樂譜，以最常見的六弦吉他為例，藉由在六條橫線上以數字表示應彈奏的指板位置，並以縱向對齊的方式來區別彈奏的先後順序，在情況允許下，一般會再與五線譜的樂符對齊，兩種譜同時呈現，如此一來節拍的資訊便不必記錄在指法譜上，降低六線譜的複雜度，如 Figure1.1[2] 所示。

知名的樂譜軟體 GuitarPro 也是採用上述形式來呈現指法譜。除此之外，因應所需演奏技巧的複雜程度或根據不同撰譜者的習慣，在表達細節上仍可能會有所差異。

doi:10.6342/NTU202403031

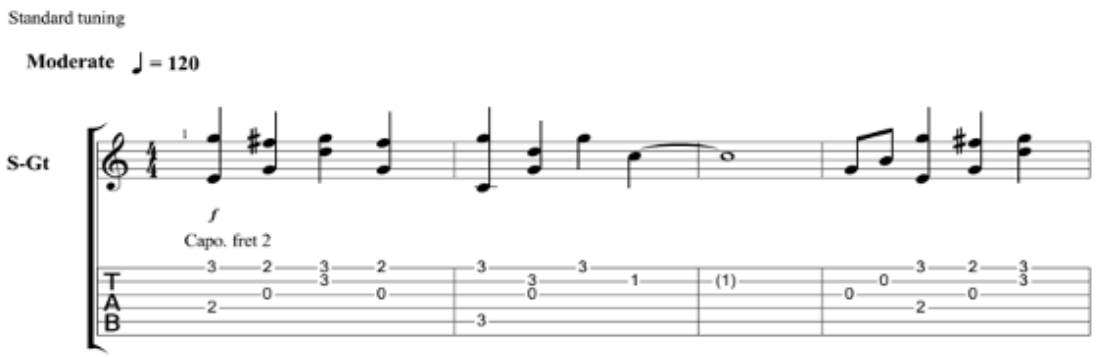

Figure 1.1: 常見的指法譜形式

1.1.2 指法譜的多樣性 (Diversity)

相較於鋼琴、直笛這些常見的樂器，吉他有一項特性，在大多情形下，一種音高可以對應到不唯一的演奏條件。演奏條件是樂器發出特定聲音時所對應的條件，可視為樂器的一種狀態。實務上常藉由樂器外部之機制 (通常為人，亦可為其他輔助器具)，改變樂器的狀態以滿足所需之演奏條件。以下以實例說明，鋼琴在標準調音的前提下，音高對應到琴鍵的關係為一對一，因此要演奏出對應的音高，其演奏條件即為按壓對應的琴鍵，此條件具有唯一性。

但吉他在標準調音的前提下，大多音高對應到弦及指板編號組合的關係為一對多，因此要演奏出對應的音高，在各條弦上，若能找到該音高對應的指板編號，則在該位置按壓該弦後彈奏，皆可演奏出對應的音高，因此其演奏條件不唯一，即具有多樣性。

Table1.1 呈現標準調音之吉他各弦在按壓各個指板下所彈奏的音高，音高以 MIDI 規格 [3] 之正整數表示，同一列代表同一條弦，同一行則代表同樣的指板編號，按壓指板位置 0 代表在不按壓琴弦的前提下撥動琴弦所對應的音高。

doi:10.6342/NTU202403031

	按壓指板位置																				
弦	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
2	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
3	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
4	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70
5	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65
6	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

Table 1.1: 在標準調音下，弦與按壓指板位置所對應之音高

1.2 研究動機與目的

在以前學習吉他的過程中，想彈奏自己有興趣的樂曲時，經常會發現找不到指法譜，但找到對應的樂譜則相對容易，然而，對於業餘愛好者來說，編排出一套洽當的指法並不是一件容易的工作，因為當時沒有時間和金錢去學習這些知識，所以只能放棄，待過一段時間後再次搜尋看看是否有人提供指法譜，這是一段令我有些沮喪的經驗。

因此在學習機器學習相關的理論與實作後，我希望建立一套系統，能夠完成將樂譜轉為指法譜的工作，透過這套系統，讓業餘的吉他愛好者能更容易取得指法譜。除此之外，若有不熟悉吉他的音樂家想用吉他呈現其作品，也能夠透過這套系統獲得指法編排上的協助。

1.3 方法概述

我們認為將樂譜轉譯為指法譜與自然語言處理中的文字翻譯任務有相似之處，因此在本節中將先回顧文字翻譯任務，再解釋樂譜轉譯的相似之處。

doi:10.6342/NTU202403031

1.3.1 自然語言處理中的文字翻譯任務

在自然語言處理中，文字翻譯是一項常見的任務，也是目前相當熱門的研究領域之一。在人類從事翻譯工作時，其難度會隨著內容浮動，事實陳述通常相對容易，而俗諺則會考驗譯者在兩種語言中的知識，諧音雙關有時候甚至會找不到洽當翻譯方式，往往需要透過註解來解釋原文脈絡。即便難度不同，同一句話大多情形下都可以有不同的說法，例如中文「一加一」和「一加上一」對大多數的人來說都可以理解成數學式中「1+1」。但也有兩句話僅僅差了一個字或詞就會產生完全不同的意義，例如英文「Do you have time?」和「Do you have the time?」。

評價翻譯的優劣，除了文法是否正確以外，通常還要比較其文字安排在該語言中是否常見，若過於罕見，還可能讓大多數的人無法理解。

目前自然語言處理針對翻譯任務較常見的做法，是先訂定語言中字詞的最小單位 (Token)，透過嵌入 (Embedding) 將這些最小單位以向量 (Vector) 表達，建立一個以上述向量作為輸出與輸入的序列模型（Sequence Model），透過大量資料進行訓練並評估其表現，選擇出一個適合承擔翻譯任務的模型。

1.3.2 樂譜轉譯為指法譜的任務

人類在說話時，除了文字順序，還有語氣、音調、重音和節奏等非語言元素，能夠提供額外的情感和語境線索，當化為文字記錄時，便只能依靠標點符號、上下文或附加的描述來理解情感和語氣。這和音樂之於樂譜的關係有相似之處，所以我們希望能在樂譜的處理上應用文字處理的方法。

我們可以將樂譜與指法譜當作兩種不同語言的文章看待。將樂譜中，同一個時間點彈奏的所有音高收集到一個集合，稱為音高集合，配合樂譜本身的順序資

doi:10.6342/NTU202403031

訊，可以形成音高集合序列，與之對應的指法譜則是指法序列。將音高集合當作是一種語言的文字，而指法則是另一種語言的文字，將這些音高集合序列轉換成指法序列，延續前述之指法譜的多樣性，一個音高集合可能對應多種指法，從這個角度看待，的樂譜轉譯任務與文章翻譯任務其實非常相似，也是本研究採用序列模型中的 Transformer 架構來完成這項任務的原因。具體如何定義這些資訊，以及模型如何與這些資訊互動，將在第三章詳述。

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

第二章　相關研究

本研究應用自然語言處理的方法來解決指法譜的轉譯問題，所以首先我們會先回顧自然語言處理方面的相關研究，再介紹與吉他指法譜相關的研究。

2.1　自然語言處理

自然語言處理主要分為自然語言理解 (Natural Language Understanding, NLU)、自然語言生成 (Natural Language Generation, NLG) 兩個面向，常見的應用涵蓋資訊檢索、文本分析、語音識別與生成、對話系統... 等等。在文字相關的應用中，目前常見的作法是透過詞嵌入 (Word Embedding)，將文字資料轉為深度學習 (Deep Learning) 模型能夠處理的資料形式，而處理這類應用最常使用的模型架構為序列模型 (Sequence Model)，以下我們將就這些研究做簡單的介紹。

2.1.1　詞嵌入 (Word Embedding)

這是一種將詞或字作映射到向量空間的方法，藉由向量的形式呈現資訊有利於作為機器學習模型的輸入與輸出，例如 One-Hot Encoding 的作法，將詞映射到一個維度大小等同於詞彙量大小的空間，每個維度代表一個詞，這項作法的缺點是向量維度太大，導致計算複雜度以及空間複雜度過高，而且無法呈現語意以及各個詞彙之間的關聯性 (各個詞彙之間的距離相同)，所以漸漸被使用較小維度的

doi:10.6342/NTU202403031

向量且考量語意以及詞彙之間關聯性 (詞義、字首、字根... 等等) 的方法取代，例如 Word2Vec[4]、Glove[5]、FastText[6] 等等。

2.1.2 序列模型

在序列模型中，RNN[7] 是最早被提出的架構之一，藉由其能夠保留先前資料的機制，來處理具有前後關係的序列資料。之後其衍生的架構 LSTM[8] 也被提出，改善了原先 RNN 長距離依賴 (Long-Term Dependencies) 的缺點。而近年來最具代表性的序列模型架構為 Transformer[9]，相較於 RNN 與 LSTM 僅能納入前文資訊的架構，其自注意力機制 (self-attention) 還能納入後文的資訊，並且藉由位置編碼 (Positional Encoding) 的方法，來加入位置資訊，進而大幅改善序列模型的表現，成為了後續重要模型 (如 BERT[10]、GPT[11]) 的基礎架構。

2.2 吉他譜轉譯與生成

目前與吉他譜關聯性較高的領域主要分為兩種，一種是樂譜轉譯，也就是在樂曲內容已經確定的前提下，產生對應的指法譜。另一種則是音樂生成，也就是除了指法譜以外，包括音樂的內容在內，都是由模型生成的創作。

2.2.1 樂譜轉譯

其中一種常見的是以音訊 (Audio) 作為輸入，並產生對應的指法譜，例如以 CNN 辨識作為主要架構，將音訊經過前處理產生頻譜圖作為輸入，最後輸出指法譜的預測資訊，並與測試資料作比對來評估表現 [12]。後續也有研究者以此研究為基準，設計了許多不同的損失函數（Loss Function）試圖改善模型的表現 [13]。

doi:10.6342/NTU202403031

其次則是以樂譜 (Sheet Music) 資料作為輸入，有研究是直接採用 Ultimate Guitar 網站上的指法譜，以 LSTM 作為基礎架構來完成樂譜轉譯的工作 [14]。有些研究則是採用 MIDI 格式的資料，或基於 MIDI 所衍生之其他格式的資料，例如 GuitarPro 格式來作為資料來源 (本研究即是採用 GuitarPro 的 gp3、gp4、gp5 作為資料來源)。有研究將樂譜與指法譜資料混合，以二維陣列的形式表示，再透過 CNN 處理，進而產生指法譜 [1]。除此之外，還有以數學方法定義 Hidden Markov Model 模型 [15–17]，不使用資料調整參數，也不與資料做比對，研究設計了對於指法難度評估的數學式，以此為基礎來設計 HMM 輸出的數值，目標是生成不同難度之指法譜，並交由專家評估結果。

2.2.2 音樂創作

這類研究屬於生成式 AI，專注於音樂創作而不是樂譜轉譯，這些生成又分為樂譜生成與音訊生成，和本研究比較相關的研究領域是樂譜生成，例如以 Transformer 為架構，樂譜作為訓練資料的樂譜生成模型 [18, 19]，雖然目的與樂譜轉譯不同，結果也大多由專家來評價，但是這些研究處理資料的方法也很值得參考。

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

第三章　問題定義

在本章，我們會先列出名詞的定義，並使用這些名詞來定義本研究的核心問題並訂定研究成果的指標。

3.1　名詞定義

- 科學音高記號 (Scientific Pitch Notation)[20]:

  對於音樂使用的音高，其組成為一個大寫英文字母加上一個整數，例如 C4、A0... 等等。大寫英文字母表示音名，數字表示該音高所在的八度。

- MIDI number[3]:

  MIDI(Musical Instrument Digital Interface) 是一個工業標準的電子通訊協定，為電子演奏裝置定義了資料格式。MIDI number 定義了科學音高記號在此通訊協定中對應的整數，其數值範圍為 0 到 127。

- 演奏單位 (Token):

  我們收集在同一時間點開始的音符所涵蓋的演奏資訊成為一個集合單位，即演奏單位。在本研究中，演奏單位之資訊僅須包含所有當下被彈奏的音高 (Pitch) 及其對應的吉他指法譜 (Tablature) 資訊。在其他研究中可能還會包含 (但不限於) 節拍 (Beat)、速度 (Tempo)、和弦 (Chord)、力度 (Dynamics)... 等資訊。因此針對本研究所需要的資訊，我們可以再將演奏單位細分成音高集

doi:10.6342/NTU202403031

合 (Pitch Set) 與吉他指法譜 (Tablature)，將在後續做更詳細地定義。

- 音高集合 (Pitch Set):

  演奏單位中 (Token) 所有被彈奏的音高所形成的集合。若以 MIDI number 描述音高，則可為如下形式 (不限於此) 表達: {60, 64, 67}，在本問題中其元素數量至少一個 (才有彈奏之必要)，由於吉他構造的限制，至多不能超過吉他弦的數量。

- 指板位置 (Fret Number)[21]:

  指板位置是根據鑲嵌在指板上的金屬條，將指板分成許多格。一般來說，在彈奏時，將不按壓琴弦彈奏的指板位置定義為 0，再根據金屬條的分割，最接近調音鈕的格子指板位置為 1，此數值在往琴身的方向為遞增的連續整數，最大值則根據吉他的規格有所不同。

- 指法譜 (Tablature)[21]:

  透過六條橫向的平行直線分別代表六條線，在線上以數字表示彈奏時須按壓個指板位置，未標示數字時則代表不得彈奏，數字間以縱向對齊的方式來表示同時彈奏，由左至右則為彈奏的先後次序。

3.2 問題定義

在這個小節中，我們會先定義問題，定義我們的輸入與輸出以及我們如何評估方法的成效。

doi:10.6342/NTU202403031

3.2.1 基本定義

對於音高集合 (Pitch Set)，我們定義其為一集合 **P**，具有以下性質:

$$1 \leq |\mathbf{P}| \leq 6, \text{ and } p \in [40, \ldots, 84], \forall p \in \mathbf{P}$$

集合中元素數值的意義為需要被彈奏的音高所對應的 MIDI number，本研究主要針對六弦吉他，最多同時彈奏六個不同音高，因此 **P** 大小上限為 6。因為我們限制吉他指板位置上限為 20，且為標準調音，因此只能彈奏 40 到 84 這個區間的音高。

對於吉他指法譜 (Tablature)，我們定義其資訊為一向量 **S**，

$$\mathbf{S} = (s_1, s_2, s_3, s_4, s_5, s_6), \text{ and } s_j \in [0, \ldots, 21], \ j \in [1...6]$$

其中 s_i 代表吉他第 i 弦的彈奏狀態，數值為 21 代表該弦為靜音 (不彈奏)，其餘數值則代表該弦必須彈奏，且該數值為彈奏時所須按壓的指板位置。

針對符合上述定義的若干個 **P** 所形成之序列 Q，我們定義序列 Q 的長度為一正整數 n，故 $Q = (\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_n)$，接著定義吉他指法譜之向量 **S** 所構成的序列 $Q = (\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_n)$，T 為 Q 所對應的指法譜序列。我們研究的問題是尋找一個方法 f，以 Q 作為輸入，令其輸出對應的 T，即 f(Q) = T。

以下將針對輸入的設計，只納入音高而不包含節奏做解釋。

我們仔細探討了編排指法譜所需要的資訊。首先，納入需要彈奏的音高是很直覺的結論，這是最重要的資訊，因為沒有需要彈奏的音高就根本沒有彈奏的必要，自然就沒有指法編排的問題需要解決。再來考量節拍、速度資訊，這兩者決定了彈奏的速度，直觀上應該會影響指法的編排，當速度越快，指法能做變

doi:10.6342/NTU202403031

換的時間就減少，理論上指法選擇就會變少，導致指法的編排會更傾向做出變動較少的選擇。但透過觀察許多表演以及指法譜的內容後，我們初步假設指法安排和節拍、速度的關聯性可能不高。首先，我們發現，在我們有觀察到的指法譜中，在節奏較慢的情形下，其編排還是都傾向最小化指法的變化，例如 Ulli Boegershausen 所演奏的 Right Here Waiting[22]，可以理解成人類在安排指法時，即便有時間上的餘裕，仍會盡可能地尋找相對容易彈奏 (指法變化較少) 的指法序列。同時我們也發現，專業的吉他手在演奏節奏較快的樂曲時，即使遇到指法變化較大的情形，也都能從容應對，例如 Tommy Emmanuel 所演奏的 Classical Gas[23]。所以雖然在節奏較慢的樂曲中，演奏者通常有較寬裕的時間來應對較大幅度的指法變化，編排依然會以較小的變化為主。然而，在節奏較快的樂曲中，如果為了完整呈現樂曲而選擇了大幅度的指板位置變化，通常只要透過演奏者的練習，就能克服這樣的問題。也就是指法的編排最優先考量的應該是能否完整呈現樂曲，接著無論節奏快慢，都會盡量降低指法的變化幅度。因此，我們先在「忽略節奏資訊」的前提下，輸入只考慮音高資訊，將根據研究的成果再決定是否納入其他資訊。

因為吉他有不同的規格與調音、輔助道具、彈奏技巧會影響音高與指法的對應關係，而我們希望能先在一個相對單純的範圍內得出一個初步成果，後續再嘗試處理更複雜的問題。因此，先以初學者入門時較為常見的條件為基準作為我們研究的範圍。

以下逐一列出本研究的問題範圍，並說明原因:

首先，我們先定義吉他的規格:

1. 以標準調音 (Standard Tuning)[24] 之六弦吉他為對象:

   標準調音之六弦吉他是最為常見且最常作為入門使用的吉他規格。

doi:10.6342/NTU202403031

2. 指板位置不超過 20:

以入門者常用的民謠吉他為例，在琴身有缺角的情況下，大多吉他的指板位置最高是落在 20 左右。雖然電吉他通常有更高的數值，若我們允許這些更高數值，我們的研究結果很可能無法適用於大多的民謠吉他規格，為了訂定一個較為泛用的問題範圍，因此我們先訂 20 為指板位置的數值上限。

3. 不使用移調夾 (Capo)[25]:

使用移調夾在抽象意義上可以說是減少最大指板位置並調高琴弦基音，會違反我們以標準調音，指板位置上限為 20 的前提。

針對一些吉他的演奏技巧，我們做以下的化簡:

1. 彈奏技巧中的搥弦，勾弦，滑弦 [26] 所產生的兩個音高，視為兩個不同 token 中的音高:

雖然這類演奏技巧所產生的聲音和一般撥弦有所不同，但聽者仍能辨別聲音的起始時間點，在一般樂譜上的紀錄也會是兩個獨立音符，而且手指也必須移動到對應的指板位置，因此我們將其視為兩個不同 token 中的音高。

2. 彈奏技巧中的推弦 [27]，僅考量起始音的資訊:

推弦技巧通常是從某個指板按壓並彈奏後，進一步將弦推彎，進而造成弦張力增加而改變音高，使其逐漸升高，過程中還有可能來回使用 (推-回復-推-回復)，來製造頻率差較小的聲音變化。整個技巧的過程中按壓的指板位置數值不變，因此我們僅考量起始位置。

3. 忽略打板 [28] 技巧資訊:

打板的聲音在物理意義上雖有音高之分，但在音樂層面的性質只是作為節拍之用，並不會特別定義演奏時必須對應的音高，並且不會訂定必須按壓的指板位置。

doi:10.6342/NTU202403031

基於上述訂定的吉他規格，我們必須限制輸入的範圍:

1. 輸入的音高必須是可以彈奏的:

   標準調音、最大指板位置為 20 所能彈奏的音高範圍為 MIDI number 的 40 到 84。若輸入的音高超出此範圍，代表不是上述規格的吉他可以彈奏的音高，故不存在任何可以對應的指法編排。基於本研究不存在對應的輸出，我們必須排除這些輸入。

2. 排除使用泛音 [29] 技巧的情形:

   泛音是比較進階的技巧，而且相對先前提到幾種演奏技巧都更為罕見，並且會導致指板位置可彈奏的音高產生分歧。因此我們將排除使用泛音技巧的情形。

3.2.2 成果評估

為了評估輸出的品質，我們主要考量兩個面向，一是該指法譜實際對應之音高集合與輸入之音高集合之間的相符程度，二是指法譜與測試資料指法譜的相似程度。輸出一份音高盡可能地正確且編排盡可能地接近人類習慣的吉他指法譜即為本研究的目標。符號定義將延續我們先前的定義，但將增加一些內容:

1. 上標 g 代表測試資料 (Ground Truth)，上標 r 代表方法 f 的輸出資料 (Result)。

2. 下標 i 代表其位於序列中的位置索引 (Index)，這部分與之前相同。但在表示指法譜時，除了位置索引，還要考量弦的編號，因此下標 i, j 代表第 i 個 token 中第 j 弦的資料。

以下將說明我們評估的指標。

doi:10.6342/NTU202403031

3.2.2.1 指法譜 0/1 正確率 (Tablature 0/1 Accuracy)

對於每個 token，若輸出結果中，六條弦的指法皆與測試資料相符，該 token 判定為正確; 反之，若存在一條弦之指法與實際資料不相符，則該 token 為錯誤。我們以以下數學式表達：

$$\text{same}(x, y) = \begin{cases} 1 & \text{if } x = y \\ & x, y \in \mathbf{N} \\ 0 & \text{if } x \neq y \end{cases}$$

$$\text{指法譜 0/1 正確率} = \frac{\sum_{i=1}^{n} \text{same}(6\,,\ \sum_{j=1}^{6} \text{same}(s_{i,j}^{g}\,,\ s_{i,j}^{r}))}{n} \tag{3.1}$$

這項指標的數值越高，代表輸出的指法編排越接近人類習慣。

3.2.2.2 音高正確率 (Pitch Accuracy)

每個 Token 都有必須被彈奏的音高，計算輸出結果與測試資料相符的音高比率。我們以以下數學式表達：

$$\text{音高正確率} = \frac{\sum_{i=1}^{n} |\mathbf{P}_i^g \cap \mathbf{P}_i^r|}{\sum_{i=1}^{n} |\mathbf{P}_i^g|} \tag{3.2}$$

這項指標的目的在於觀察我們的模型是否能夠識別音高資料的意義，若是模型在此指標的表現良好，但在前兩個指標的表現較差，代表模型能夠反映需要被彈奏的音，卻未從我們設計的方法學到人類的指法編排。

3.2.2.3 音高誤報比率 (Pitch False Alarm Ratio)

每個 Token 都有必須被彈奏的音高，計算輸出結果中存在但並未出現在測試資料的音高比率。我們以 $\overline{\mathbf{P}_i^g}$ 代表 $\mathbf{P}_i^g$ 之補集，也就是從所有可以被吉他彈奏的音

doi:10.6342/NTU202403031

高中，排除樂譜上記載應該被彈奏的音高後，剩餘的音高所形成的集合。接著以以下數學式表達：

$$\text{音高誤報比率} = \frac{\sum_{i=1}^{n} |\overline{\mathbf{P}_i^g} \cap \mathbf{P}_i^r|}{\sum_{i=1}^{n} |\mathbf{P}_i^g|} \tag{3.3}$$

這項指標的目的在於觀察我們的模型預測出多少不應該被彈奏的音。

doi:10.6342/NTU202403031

第四章　研究方法

在本章，我們將先介紹我們使用的資料集以及針對這個資料集所做的前處理，再來會介紹我們所使用的模型: 輸入、輸出的設計，模型架構... 等等。

4.1　資料集

在這節中，我們會先介紹資料集的基本資訊，以及我們如何選取、過濾這些資料。

4.1.1　資料集簡介:dadaGP

本次實驗我們使用 dadaGP 資料集 [30]，共包含 26,181 筆樂譜資料，格式包含 gp3、gp4、gp5(以下統稱 gp 檔案)，資料集作者也替這些資料歸類到 739 個類別。

4.1.2　檔案過濾

本資料集作者有提供其自製的資料 encoder、decoder(此部分並不是指 Transformer 的 encoder、decoder)，作者將許多原始資料經過 encoder 做處理的檔案，以及前者再透過 decoder 做還原的檔案，以及使用 PyGuitarPro API 轉檔所產

doi:10.6342/NTU202403031

生的檔案都一併放在資料集內，以下為例子之一。

```
$ tree path/to/dadaGP/DadaGP-v1.1/A/Alice
.
|-- Alice - Arabic Heavy.gp3
|-- Alice - Arabic Heavy.gp3.gp2tokens2gp.gp5
|-- Alice - Arabic Heavy.gp3.pygp.gp5
--- Alice - Arabic Heavy.gp3.tokens.txt
```

這些資料中僅 Alice - Arabic Heavy.gp3 是原始資料，因此其他的資料必須被排除。實際上的做法是撰寫 Python 腳本，根據原始 dataset 的檔案結構複製一份，必須被排除的資料則不複製。

4.1.3 使用 PyGuitarPro 套件過濾資料

由於資料集作者的 encoder、decoder 不符合本研究需求，我們必須透過其他方式讀取資料，我們使用開源套件 PyGuitarPro[31] 來讀取資料，每個 gp 檔案可被此套件讀取為一個 song 類別的物件。以下程式碼將呈現物件的部分資料結構，程式碼中的 song_obj 即為 song 類別之物件。

```
for t, track in enumerate(song_obj.tracks):
    #track.channel.instrument: 樂器種類
    #track.strings: 弦的數量
    #track.strings: 可取得各弦基本音高
    #track.offset: capo 夾位置
    for m, measure in enumerate(track.measures):
        for v, voice in enumerate(measure.voices):
            for b, beat in enumerate(voice.beats):
```


doi:10.6342/NTU202403031

```
9                   #beat.hasHarmonic: 記錄是否由泛音技巧彈奏
10                  for n, note in enumerate(beat.notes):
11                      #note.type: 類型(連音/休止/打板/彈奏)
12                      #note.string: 彈奏的弦
13                      #note.value: 按壓的指板位置
```

在這些迴圈中，以 track 為單位，根據我們問題定義中的輸入限制做篩選，符合條件的 track，會以獨立的 json 檔案作為中介格式紀錄其指法譜，採用如下結構: {"$measure_0$": $[token_0, token_1, ...]$, "$measure_1$": $[token_0, token_1, ...]$, ...}，其中 measure 指的是音樂中的各個小節，而 token 的內容為 $[[string, fret]_0, [string, fret]_1, ...]$，每組 [string, fret] 資料記錄了被彈奏的弦及其按壓的指板位置，未存在於記錄的弦即代表不彈奏，因此每個 token 至少包含 1 組 [string, fret] 資料，至多存在 6 組。note 只採計彈奏類型。連音並不是彈奏，只是樂譜上記載音符長度的資訊，而打板技巧與指板上的指法無關，休止則是沒有彈奏行為。

4.2 前處理

在這節中，我們會敘述我們如何將以 track 為單位，獨立儲存的中介格式合併成訓練、驗證、測試的資料集。

4.2.1 資料集分割

我們讀取所有在上一節完成的 track，依照每個 track 所彈奏之最高指板位置 (0 到 20，不考慮弦) 作為索引，建立出各個索引的 track 清單，針對每個索引，以

doi:10.6342/NTU202403031

track 為單位，以 8:1:1 的比例隨機分配到訓練、驗證、測試三個資料子集。以最高指板位置作為分割依據，是希望在三個資料子集的資料分布不要差異過大，例如可以避免產生訓練集資料的指板位置介於 0~20，但驗證集、測試集只集中在 0~5 這樣的情形。

4.2.2 截斷 (Truncation) 預處理

針對每個 track，我們必須先進行截斷，再寫入個自所屬的資料集檔案。

原則上和自然語言處理的作法相同。不過由於資料長短差異過大，各個 track 的序列長度介於 10^0 到 10^4 的範圍，一開始在程式執行時才做截斷，導致執行時經常發生 GPU 記憶體不足的問題，原因是如果 batch 是以 track 為單位做資料選取，接著再做截斷，有可能剛好都選到一些長度較長的 track，進而使該 batch 的資料量過大。

為了讓所使用的設備 (RTX-2080Ti) 能順利運作，在程式執行前就會預先將各個 track 作截斷，形成一定長度範圍內的子序列，再以這些子序列為單位組成 batch 就可以避免記憶體不足的問題，同時我們會紀錄這些序列的來源資訊 (所屬 gp 檔案、所屬 track 以及其為截斷後的第幾個子序列)，以備未來需要檢視資料來源之用。

訓練集的部分，我們以長度 126 為單位進行截斷，若截斷後之子序列長度未達 14 則捨棄之。長度上限的設定是希望模型能夠在考量足夠長資訊。少於 14 則捨棄是由於整體資料量龐大，我們希望能縮短模型訓練的時間，較短的序列其資訊含量相對較少，捨棄後的影響應該相對較小而為之。

至於在驗證集以及測試集，也是以長度 126 為單位進行截斷，所有不足 126 的子序列皆捨棄，此作法一方面也是為了加速整體流程，另一方面也是想讓評估

doi:10.6342/NTU202403031

指標的數學設計較為直觀，利於程式碼的實現。

根據上述做法執行後，序列數量的統計資料如下: 訓練集為 161923 筆，驗證集為 18104 筆，測試集為 17628 筆。

我們將這三個資料子集分別以 json 檔案儲存。

4.3 模型

在這一節中，會先介紹輸入與輸出的嵌入方法與位置編碼，再介紹本研究基於 Transformer 發想的模型架構，最後會再介紹評估指標。

4.3.1 輸入、輸出之處理

在這個小節中，我們將敘述我們呈現資料的方法，也就是嵌入 (Embedding) 的設計: 如何以向量呈現輸入的音高集合以及輸出的指法譜。隨後便會介紹我們為了應用 Transformer 模型應用的位置編碼 (Positional Encoding)。

4.3.1.1 嵌入 (Embedding) 設計

自然語言處理中，經常使用詞嵌入 (Word Embedding)，使用維度遠小於字典大小的向量來表示文字資訊 [32, 33]，本研究也將使用這樣的概念來描述資訊並作為模型的輸入與輸出。

我們先以輸出，也就是指法譜的角度切入，如同我們在問題定義中所提及的，每條弦有 22 種狀態，包含按壓位置為 0~20 的條件下被彈奏共 21 種狀態，再加上 1 種不彈奏的狀態。吉他總共有 6 條弦，因此共有 $6 \times 22 = 132$ 項資訊，因此我們先考慮每個 token 使用 132 維的向量，每 22 維為一條弦的資訊。132 維中

doi:10.6342/NTU202403031

的 6 組 22 維度，依序代表第 1 弦到第 6 弦。每 22 個維度中，前 21 維分別對應到彈奏按壓位置的資訊，第 22 維則代表不彈奏的情形，該弦所處狀態代表的維度值為 1，其餘皆為 0。我們使用 132 維元素皆為 1 的向量作為 BOS(Begin of Sentence) token，原因是一般的 token 恰好會有 6 個維度為 1，其餘為 0，相較之下這樣的設計能夠和一般的 token 有比較大的區別；至於 padding token 則是每條弦的最後一個維度為 1，其餘為 0，也就是代表每條弦都不彈奏的情形。

接下來考慮輸入，也就是音高集合 (Pitch Set) 的資訊，我們採用 MIDI[3] 的規格：共 128 種可能，與上述 132 數值相近，因此我們也採用 132 維度的向量，最後 4 維固定是 0，其餘維度則視其代表的音高是否有被彈奏，有被彈奏為 1，其餘為 0。這部分用不到 BOS，但需要 EOS 和 padding，由於這兩者和完全沒有彈奏的音高集是相同意義的，所以我們採用全為 0 的向量做為代表。

4.3.1.2 位置編碼 (Positional Encoding)

我們讀取前處理過的 json 格式來得到音高集合序列以及指法譜序列，透過上一個小節的嵌入方法，可將這些資訊轉為向量表示，但我們還需要位置資訊，所以我們再加入位置編碼的資訊，利於我們應用 Transformer 模型來解決問題。我們採用 Transformer 模型被提出的論文中 [9] 的數學方法：

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{4.1}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{4.2}$$

其中 pos 代表 token 的位置，i 代表元素所在的維度，d_{model} 代表向量的維度 (在本研究中的數值為 132)。根據4.3.1.1的 Embedding 方法，將資料轉為向量表示後，先將向量乘上 $\sqrt{d_{model}}$，再加上 Positional Encoding 的資訊，即為模型的輸入。

doi:10.6342/NTU202403031

4.3.2 模型基本架構

在本小節中，我們將介紹基於 Transformer[9] 的模型架構，接著介紹我們實作用的套件與訓練方法，最後還有我們設計的後處理方法。

4.3.2.1 架構

如1.3的部分所述，我們將轉譯指法譜以語言翻譯的邏輯來處理，因此我們採用 Transformer[9] 作為原型，其架構如 Figure 4.1所示。我們希望模型在 Decoder 能夠輸出包含 Positional Encoding 資訊的指法譜向量序列，便可以透過扣除 Positional Encoding 的資訊，再根據 Embedding 的流程進行反運算，得出預測的指法。

doi:10.6342/NTU202403031

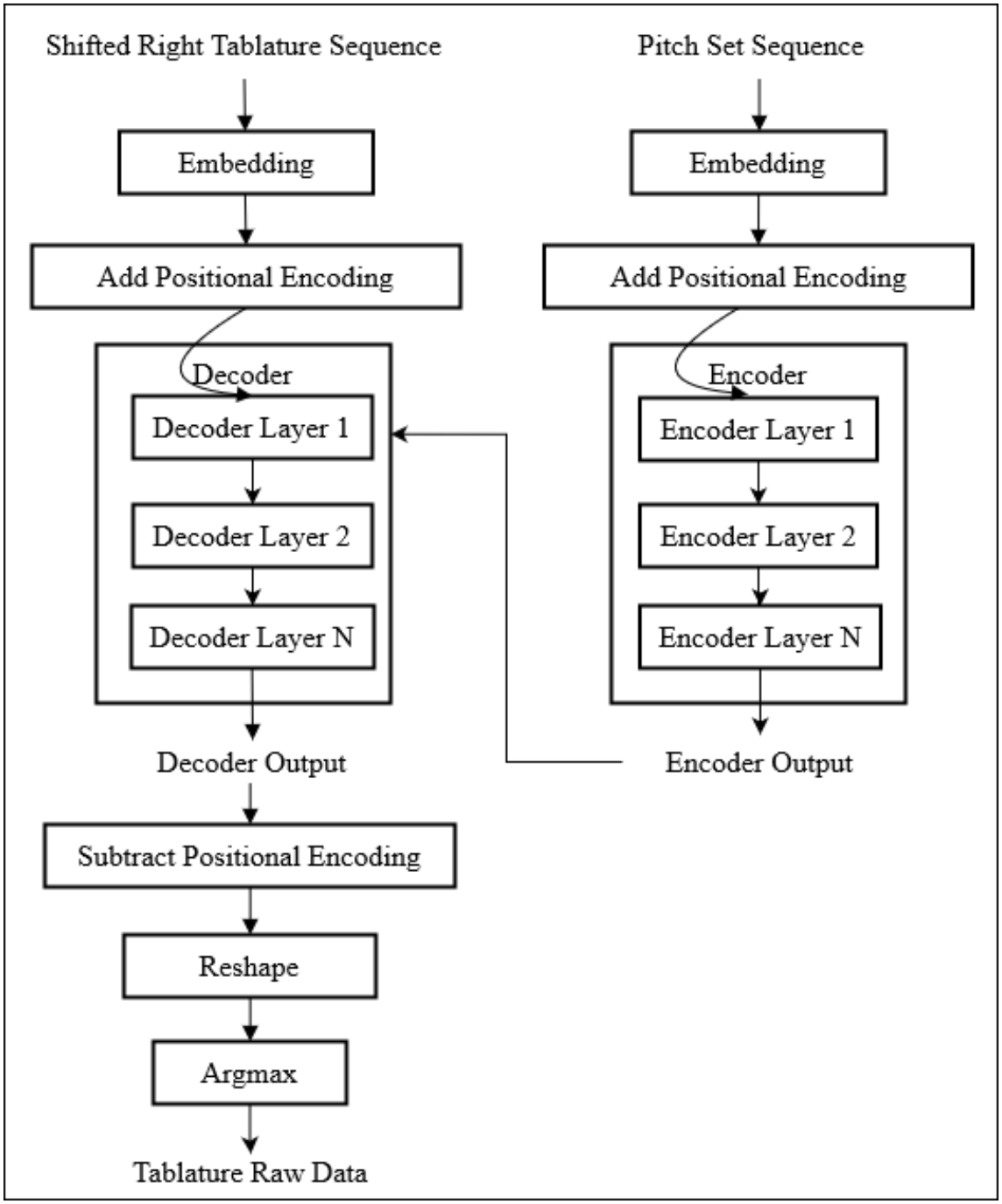

Figure 4.1: 基於 Transfomer 設計的模型架構

doi:10.6342/NTU202403031

4.3.2.2 實作與訓練

我們採用 PyTorch 套件中的 torch.nn.Transformer[34] 來實作模型，並且採用 Learning Rate Scheduling[9]:

$$lrate = factor \times LRS \tag{4.3}$$

$$LRS = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \tag{4.4}$$

我們採用均方差 (MSE, Mean Squared Error) 作為損失函數，計算指法譜經過 Embedding 與 Positional Encoding 後的資料與 Decoder Output 之間的均方差，來訓練模型。訓練過程中我們也使用 tgt_mask 來訓練 decoder 逐步推理的能力，使用 src_key_padding_mask、tgt_key_padding_mask，針對 padding token 做 mask。

4.3.2.3 後處理

我們問題定義中的輸入包含了彈奏的音高資訊，我們其實可以檢查 Result 中所彈奏的音高是否與 Inputs 相符，藉此刪除明顯錯誤的資訊。例如實際需要被彈奏的音高集是 {40, 45, 50}，而預測結果的音高集為 {40, 45, 50, 55}，則我們可以將預測結果中，彈奏 55 的那條弦改為不彈奏，我們將這個程序稱為 Pitch False Alarm Mask，整體架構如 Figure 4.2所示。一開始沒有直接實作這個架構是因為不太確定能否成功以 GPU 的平行運算實現。我們針對所有的 token，每次只檢查一條弦所彈奏的音高，收集發生錯誤的 token 索引，接著將這些位置的 token 中該弦的資訊改為不彈奏。在使用平行運算，記憶體充足的前提下，時間複雜度為 O(1)。

doi:10.6342/NTU202403031

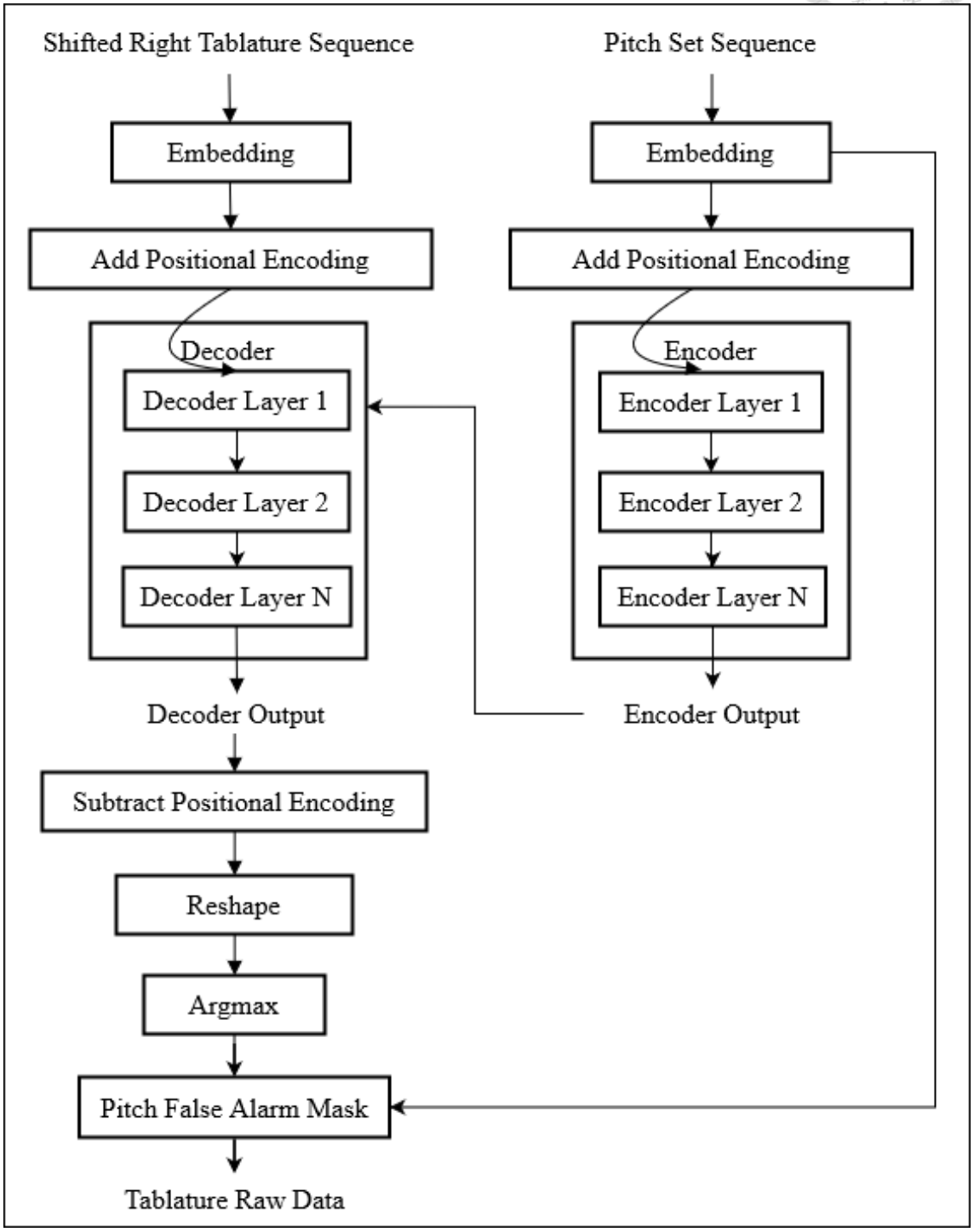

Figure 4.2: 加上後處理的模型架構

doi:10.6342/NTU202403031

4.4 評估指標

除了沿用我們在問題定義章節中提及的三項評估指標，我們新增了一項指法編排相似度的指標，將於該小節中詳細說明。

4.4.1 指法譜 0/1 正確率 (Tablature 0/1 Accuracy)

對於每個 token，若輸出結果中，六條弦的指法皆與測試資料相符，該 token 判定為正確; 反之，若存在一條弦之指法與實際資料不相符，則該 token 為錯誤。我們以以下數學式表達：

$$\text{same}(x, y) = \begin{cases} 1 & \text{if } x = y \\ & \qquad x, y \in \mathbf{N} \\ 0 & \text{if } x \neq y \end{cases}$$

$$\text{指法譜 0/1 正確率} = \frac{\sum_{i=1}^{n} \text{same}(6\ ,\ \sum_{j=1}^{6} \text{same}(s_{i,j}^{g}\ ,\ s_{i,j}^{r}))}{n} \tag{4.5}$$

這項指標的數值越高，代表輸出的指法編排越接近人類習慣。

4.4.2 音高正確率 (Pitch Accuracy)

每個 Token 都有必須被彈奏的音高，計算輸出結果與測試資料相符的音高比率。我們以以下數學式表達：

$$\text{音高正確率} = \frac{\sum_{i=1}^{n} |\mathbf{P}_i^g \cap \mathbf{P}_i^r|}{\sum_{i=1}^{n} |\mathbf{P}_i^g|} \tag{4.6}$$

這項指標的目的在於觀察我們的模型是否能夠識別音高資料的意義，若是模型在此指標的表現良好，但在前兩個指標的表現較差，代表模型能夠反映需要被彈奏

doi:10.6342/NTU202403031

的音，卻未從我們設計的方法學到人類的指法編排。

4.4.3 音高誤報比率 (Pitch False Alarm Ratio)

每個 Token 都有必須被彈奏的音高，計算輸出結果中存在但並未出現在測試資料的音高比率。我們以 $\overline{\mathbf{P}_i^g}$ 代表 $\mathbf{P}_i^g$ 之補集，也就是從所有可以被吉他彈奏的音高中，排除樂譜上記載應該被彈奏的音高後，剩餘的音高所形成的集合。接著以以下數學式表達：

$$\text{音高誤報比率} = \frac{\sum_{i=1}^{n} |\overline{\mathbf{P}_i^g} \cap \mathbf{P}_i^r|}{\sum_{i=1}^{n} |\mathbf{P}_i^g|} \tag{4.7}$$

這項指標的目的在於觀察我們的模型預測出多少不應該被彈奏的音。

4.4.4 指法編排相似度 (Slot Accuracy)

每個 token 具有 6 條弦的指板資訊，可以視為每個 token 有 6 個 slots 資料必須進行預測，slot 的預測結果與實際資料相符則判定為正確，也就是每條弦的預測正確與否是個別計算的。將上述計算方式用以下數學式表達：

$$\text{指法編排相似度} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{6} \text{same}(s_{i,j}^g, s_{i,j}^r)}{6n} \tag{4.8}$$

指法譜 0/1 正確率，我們希望更細部的去區分到底一個 token 有多少比例的輸出是與實際資料相符，假設輸出結果出現了極端情形: 每個 token 總是恰好錯一條弦，則指法譜 0/1 正確率為 0，在指法編排相似度則為 0.8。我們認為綜合這兩項指標能讓我們更清楚地觀察輸出結果。

doi:10.6342/NTU202403031

4.5 不採用 rule-based 的原因

剛開始研究時也有考慮過 rule-based 的方法，而看過的相關研究中最接近的大概就屬以 HMM 輸出不同難度吉他譜的研究 [16]，該研究的數學方法已經設想得很周到，可以作為 rule-based 方法的參考對象，但仍有改進空間，例如在指板位置差距上的難度衡量，例如要同時按第 1 格與第 4 格的指法組合，可以說是相當難的指法，但同樣的格數差距，放到第 9 格和第 12 格就會相對簡單，因為指板間距會隨著指板位置上升逐漸下降，第 1 格與第 4 格的指法組合手指需要橫跨的距離相較第 9 格與第 12 格的組合來得大，但我們又應該做什麼樣的數值操作來平衡這樣的差異呢? 簡單的作法可能是做一些線性轉換，讓高格數的橫跨難度較低，透過不斷調整這項線性轉換的參數，尋找表現較好的一組參數。除此之外我們也考量到，也許不同弦的按壓難度也有分別，需要的按壓的弦數量就算一樣，但如果是跨度大的組合可能會是食指、小指的組合才能應對，這時候難度比較高，跨度小就可能是食指、中指的組合就可以應對，難度就比較小，但如果跨度小但按壓弦數較多，和跨度大但按壓弦數較小，這兩者之間又該如何評估? 也許都可以透過 trial and error 的方式來取得一個結果，但可能曠日費時也不見得能得到足夠好的結果。現在既然有整理好的資料集，也有成熟的機器學習技術以及足夠的運算資源支撐，我們相信這樣的方法會較上述的 trial and error 更有效率且更有機會取得成果。

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

第五章　實驗

在本章，首先先介紹我們實驗的方向，試著針對本研究定義的問題 (3.2) 訓練出一個表現盡可能好的模型。此外，我們也尋找了相關研究中，與本研究定義的問題較為接近的發表 [1]，試著與其進行比較。

5.1　實驗設計

我們的第一步是先訓練出一個基於 Transformer 架構的基準模型 (Baseline Model)，如果能在其中一項評估指標上取得 50% 左右的表現，我們便以此為基準模型，持續調整並改善表現。

5.1.1　基準模型設計

由於並沒有找到與我們問題定義接近，且使用相同模型架構的研究可以參考，我們首先參考 PyTorch 中的 Transformer[34] 預設參數，設計我們的第一個 baseline model 參數量，以下將說明我們認為相對重要的參數設定。

- 輸入與輸出向量的維度，預設為 512，但我們的 embedding 設計是 132 維度，因此設為 132。
- Attention head 的數量，預設是 8，其限制為必須為 d_model 的因數，因為會

doi:10.6342/NTU202403031

平均分配維度的資訊給各個 attention head 處理。在我們的研究中，因為問題定義的限制範圍，所以輸入的音高會產生變化的維度在 40~84 之間，但指法的部分則是全部維度都有可能產生變化，所以我們希望將 attention head 的關注的維度切分得更細，但同時又不希望參數量過多導致訓練時間過程，因此先選擇了 12 作為基準模型的 attention head 數量。

- Encoder 與 Decoder 個別的層數，預設是 6，因為 Transformer 最初是處理自然語言問題的模型，但本研究的問題應該會比自然語言簡單一些，因此我們下調到 4。

- Feedforward network 的維度，預設是 2048，對於這部分我們想先採用預設值，在後續的實驗再尋找適合的設定。

其餘未提及的設定並不是本研究關注的變因，我們皆採用預設設定。

5.1.2 尋找表現較佳的參數量

有了上述基準模型後，我們主要調整 Attention head 的數量、Encoder 與 Decoder 的層數、Feedforward network 的維度，試著我們的設備可以完成，且在可以接受的執行時間中 (1 週內)，尋找表現較佳的參數量。

5.1.3 訓練與挑選模型

在最一開始的實驗中有遭遇訓練無法收斂的問題，經過多次嘗試，以(4.4)的 warmup_steps=40，(4.3)的 factor=0.0015 之設定進行訓練可以成功收斂。我們便以此作為設定來訓練模型，最大執行 epoch 數設定在 200 以上，如果發現模型在驗證 (Validation) 中發生表現逐漸下降的情形，我們便可能提前終止訓練。若發現模

doi:10.6342/NTU202403031

型的表現在結束前其實還是有些微地提升，我們會繼續觀察驗證的變化，直至訓練滿一週左右終止訓練。

5.2 與已發表研究的比較

作為比較對象的已發表研究 [1] 也是使用 dadaGP 資料集來訓練一個基於 CNN 架構，能將樂譜轉為指法譜的模型，不過處理的問題範圍以及資料使用方法和我們稍有不同，主要為以下三點:

第一點，其最大指板位置限制在 24，本研究則限制在 20，這部分僅需對上一章設計的嵌入方法作些微調整，即可應對: 指法譜改為總共 162 維度，每條弦分配到 27 個維度，包含 0 到 25 的指板位置狀態加上靜止狀態，共 27 種狀態。因此輸入也改為 162 維度的向量，但依舊是以 MIDI number 對應的維度來呈現音高資訊。

第二點，研究訓練資料僅使用 dadaGP 資料集中 5% 的資料，原文為「Data in the system was represented in a binary format (described in the next paragraph) that generated multiple instances per piece, making the entirety of the dataset practically unusable because of its large size. Therefore, a 5% part of the entire dataset was employed, in which a random 5% of pieces from each folder in the DadaGP was included.」。我們推測是因為其採用的模型架構與定義輸入資料格式導致使用全部資料會將模型訓練時間拉得太長。但本研究的模型應該不會有這樣的問題，所以本研究會採用所有符合問題限制的資料來訓練模型，並不會另外再篩選，因為能夠使用較多資料來做訓練也是本研究所使用方法的優勢。

第三點，該研究有設計資料增強 (Data Augmentation) 方法，所以有不採用資料增強的實驗結果作為對照組。我們認為本研究能夠採用的資料足夠多，我們打算先不採用這類方法，觀察本研究和該研究的表現比較之結果再作考慮。

doi:10.6342/NTU202403031

基於以上幾點，我們規劃先就本研究之問題範圍，嘗試不同的參數設定，在這之中挑出表現最好的設定，以此為雛形，調整成能夠執行與比較對象相同的問題範圍，並且將納入更高指板位置的資料進行訓練與測試，比較兩者的表現。

doi:10.6342/NTU202403031

第六章　結果與分析

在本章，主要呈現我們實驗的結果以及對於實驗結果的解釋。

6.1　結果

Table 6.1呈現我們的模型規格；Table 6.2呈現在模型訓練時所做的驗證結果，也就是我們挑選模型的依據；Table 6.3呈現透過驗證挑選出的模型在測試集的表現，所有模型訓練過程中的損失變化以及驗證結果的變化趨勢圖，由於數量較大，一併置於附錄 A 中展示。

Model	Post-processing	nhead	dim_feedforward	num_decoder_layers num_encoder_layers
#1	False	12	2048	4
#2	True	12	2048	4
#3	True	12	2048	4
#4	True	12	2048	8
#5	True	12	4096	4
#6	True	12	4096	4
#7	True	12	8192	4
#8	True	6	2048	4
#9	True	12	2048	2
#10	True	12	2048	4
#11	True	22	2048	4
#12	True	12	2048	6

Table 6.1: 模型規格

doi:10.6342/NTU202403031

Model	Total Epoch	Best Epoch	Slot Accuracy	Tablature 0/1 Accuracy	Pitch Set Accuracy
#1	290	217	0.8752	0.7346	0.9074
#2	250	194	0.8925	0.7412	0.9043
#3	400	375	0.8982	0.7578	0.9164
#4	400	397	0.8309	0.5592	0.7459
#5	306	282	0.9086	0.7820	0.9403
#6	350	347	0.9118	0.7900	0.9457
#7	350	340	0.9122	0.7833	0.9408
#8	350	310	0.8939	0.7489	0.9086
#9	200	77	0.7389	0.3749	0.5274
#10	350	334	0.8978	0.7557	0.9160
#11	350	275	0.8628	0.6381	0.8255
#12	350	335	0.8278	0.5454	0.7384

Table 6.2: 驗證 (Validation) 結果

Model	Slot Accuracy	Tablature 0/1 Accuracy	Pitch Set Accuracy	Pitch False Alarm Ratio
#1	0.8776	0.7336	0.9106	0.0803
#2	0.8934	0.7397	0.9062	0
#3	0.8991	0.7543	0.9185	0
#4	0.8327	0.5581	0.7472	0
#5	0.9087	0.7771	0.9394	0
#6	0.9118	0.7848	0.9450	0
#7	0.9121	0.7791	0.9413	0
#8	0.8946	0.7457	0.9121	0
#9	0.7411	0.3765	0.5316	0
#10	0.8979	0.7508	0.9175	0
#11	0.8661	0.6396	0.8278	0
#12	0.8297	0.5446	0.7401	0

Table 6.3: 測試 (Testing) 結果

6.2 分析

本節主要針對實驗的結果做討論，在此先針對數據做些說明，#1 是我們第一個訓練過程較完整，成功收斂的模型，同時也是所呈現的結果中唯一沒有加入後處理的模型。因為後來成功實現了藉由 GPU 平行處理的後處理程序，所以後續實驗都納入了後處理，因此 #1 是作為後處理效益的對照組。

在後續的實驗我們發現 350~400 間更佳表現的模型次數很少，即使有所改善，數值上的差異也很小，所以將後續訓練流程的最大 epoch 數訂在 350，讓我們的實驗可以更快完成。因此我們重新以與 #1 相同規格的參數，並加入後處理，訓練了 #10 作為參數調整實驗的對照組。

doi:10.6342/NTU202403031

我們主要以 Table 6.1和 Table 6.3的數值做討論，同時也會另行整理主要觀察的資料，以表格呈現在各個小節中。

6.2.1 後處理的效益

dim_feedforward: 2048, nhead: 12, num_encoder_layers: 4, num_decoder_layers: 4

Model	Post-processing	Slot Accuracy	Tablature 0/1 Accuracy	Pitch Set Accuracy	Pitch False Alarm Ratio
#1	False	0.8776	0.7336	0.9106	0.0803
#2	True	0.8934	0.7397	0.9062	0
#3	True	0.8991	0.7543	0.9185	0

Table 6.4: 後處理效益之比較

#2 與 #3 為參數量與 #1 完全一致，除了訓練的 epoch 數量不同以外其餘設定皆相同。由 #1 和 #2 在測試的結果發現，雖然 #2 在 Pitch Set Accuracy 的表現較差一些，但在 Slot Accuracy 和 Token 0/1 Accuracy 明顯領先。這樣的表現符合我們後處理的作法，因為我們是將預測結果中彈奏不正確音高的弦改為不彈奏，對於 Pitch Set Accuracy 完全沒有影響，但可以改善 Slot Accuracy 和 Token 0/1 Accuracy。這點從驗證的圖表也可以看出來，Figure 6.1中 Pitch Set Accuracy 在 epoch 80 左右開始，在數值上是持續領先 Slot Accuracy，在加入後處理的 Figure 6.2以及 Figure 6.3中可以注意到，兩者的數值明顯較為接近，一直 Figure 6.3中的 epoch 270 左右才又拉開一些差距，但明顯這樣的差距是小於 Figure 6.1所看到的。由於我們採用 GPU 平行處理的來完成後處理，所以時間成本非常低，在訓練過程中，每一個 epoch 的驗證時間大概只多花了 30 秒 (原先是 12 分鐘左右)。

6.2.2 Self Attention 次數之影響

我們將 encoder 以及 decoder 的層數設為一致，我們以 #10 為對照組，#9、#12、#4 為實驗組做比較，可以發現 encoder 和 decoder 為 4 層時表現最好，其他的表現都明顯較差。不過比較有趣的是，#9 的表現是隨著訓練次數顯著下降

doi:10.6342/NTU202403031

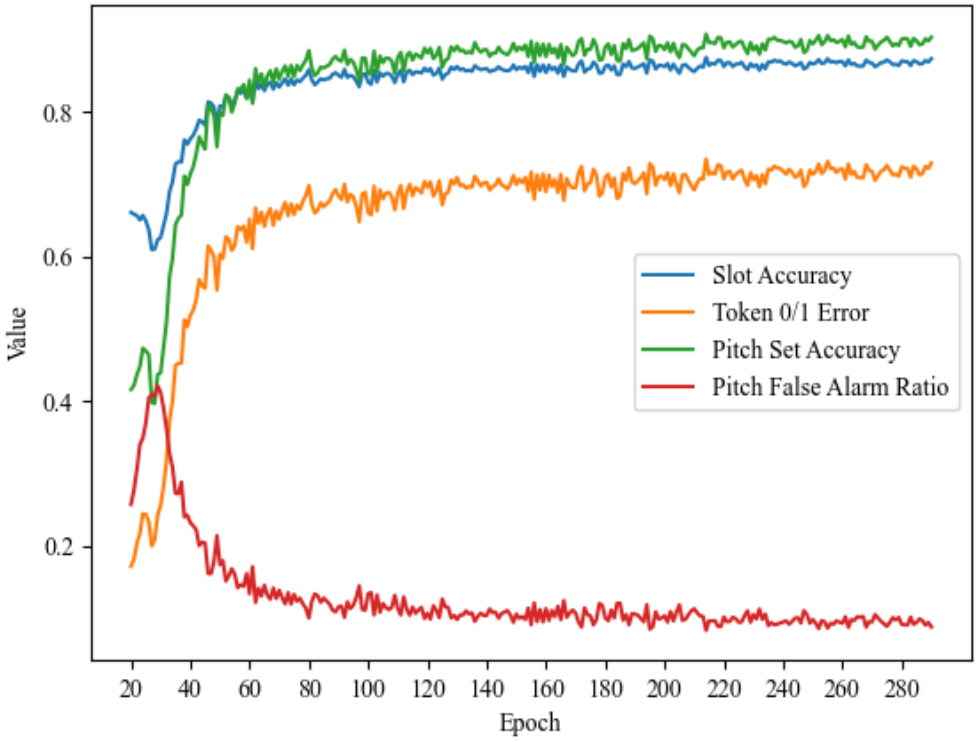

Figure 6.1: Validation Progress of Model#1

doi:10.6342/NTU202403031

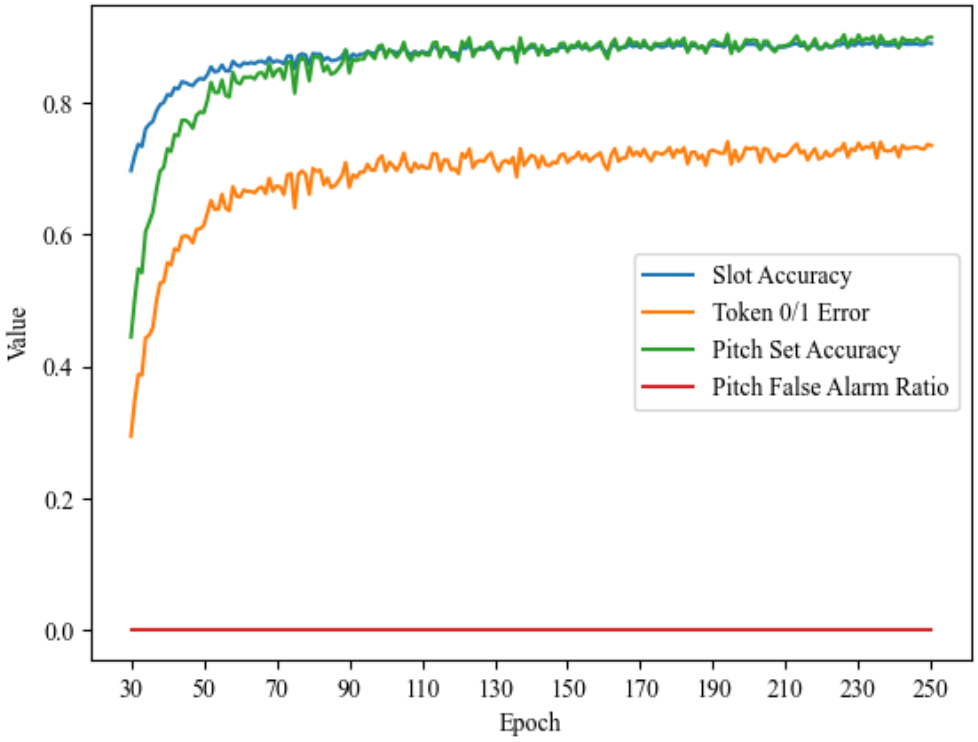

Figure 6.2: Validation Progress of Model#2

doi:10.6342/NTU202403031

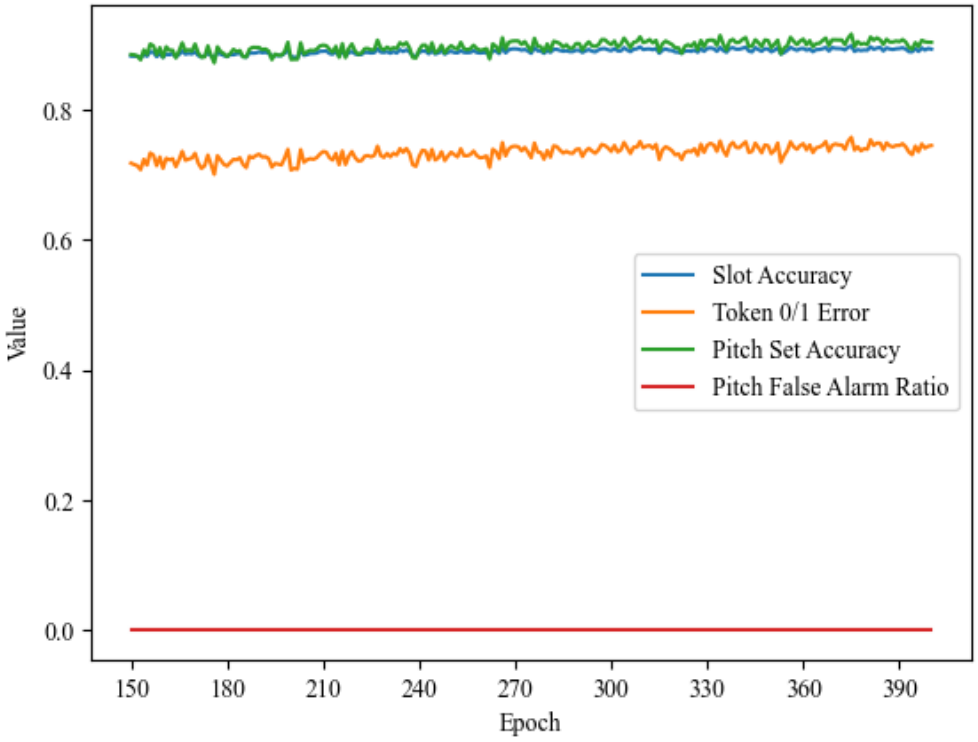

Figure 6.3: Validation Progress of Model#3

doi:10.6342/NTU202403031

Post-processing: True, dim_feedforward: 2048, nhead: 12

Model	num_encoder_layers num_decoder_layers	Slot Accuracy	Tablature 0/1 Accuracy	Pitch Set Accuracy
#9	2	0.7411	0.3765	0.5316
#10	4	0.8979	0.7508	0.9175
#12	6	0.8297	0.5446	0.7401
#4	8	0.8327	0.5581	0.7472

Table 6.5: num_encoder_layers, num_decoder_layers 之比較

(Figure A.8)，在 #4 的表現則看起來是隨著訓練次數微幅上升 (Figure A.8)，不過因為參數量增加導致，訓練耗時也大幅提升，我們並沒有去探索再之後的變化，以成本面作為考量還是 #3 的參數量在訓練時間和表現上都較為良好 (Figure A.6)。

6.2.3 Attention Head 數量之影響

Post-processing: True, dim_feedforward: 2048, num_encoder_layers: 4, num_decoder_layers: 4.

Model	nhead	Slot Accuracy	Tablature 0/1 Accuracy	Pitch Set Accuracy
#8	6	0.8946	0.7457	0.9121
#10	12	0.8979	0.7508	0.9175
#11	22	0.8661	0.6396	0.8278

Table 6.6: nhead 之比較

PyTorch 套件限制 Attention Head 數量必須是 d_{model} 之因數，才能平均分配維度資訊給各個 Attention Head。本研究中 $d_{model} = 132 = 2^2 \times 3 \times 11$，我們選用了 6、12、22 的數量來比較，如 Table 6.6所呈現，其中 #8 與 #10 並無顯著差異，但 #11 卻有表現大幅下降的趨勢，因此我們認為 12 應該是本研究問題範圍內較適合的參數量。

6.2.4 Feedforward Dimension 之影響

我們以 #3 作為比較基準，觀察 #5~#7 的表現，發現更高的數值有較佳的表現，為了有一個較為平等的比較基準，便以 #3 的規格重新訓練了 #10，並且去除訓練過程中機器故障導致提前中斷的 #5，因此比較的結果如 Table 6.7所呈現。

doi:10.6342/NTU202403031

Post-processing: True, nead: 12, num_encoder_layers: 4, num_decoder_layers: 4.

Model	dim_feedforward	Slot Accuracy	Tablature 0/1 Accuracy	Pitch Set Accuracy
#10	2048	0.8979	0.7508	0.9175
#6	4096	0.9118	0.7848	0.9450
#7	8192	0.9121	0.7791	0.9413

Table 6.7: dim_feedforward 之比較

我們可以發現，維度為 4096 的表現明顯較 2048 以及 8192 好，維度 8192 雖然比起 2048 較好，但因為其參數量過大，導致我們在進行驗證時，必須把原先慣用的 batch size 減半才能讓我們 GPU 的記憶體用量維持在滿溢邊緣，而 batch size 減半意味著整體運算時間加倍，在每個 epoch 必須執行訓練與驗證的情況下，一個小時大概只能跑不到 2 個 epoch，因此以我們的設備實務層面來說 8192 有顯著的缺點。

6.2.5 輸入資訊探討

從最好的結果 (#6) 來看，我們的指法相似度高達 0.9，這驗證了我們在3.2中的假設：「節奏資訊對於指法安排之影響可忽略」是合理的。根據本研究的結果，可以得知將樂譜轉譯為指法譜的工作，在忽略節奏資訊的情況下也能有不錯的表現。

6.3 與已發表研究的比較

在 A Machine Learning Approach for MIDI to Guitar Tablature Conversion[1] 這份研究中，我們比較的對象是這份研究 Table 2. 的 Guitar-only training - Guitar-only test 和 Augmented training - Guitar-only test 這兩項的表現。

首先，我們解釋一下這份研究中提及的名詞，tablature frame 和本研究定義的 token 中的指法譜部分是相同的定義，所以其中 match 列，sum 行的數值，也就

doi:10.6342/NTU202403031

是 token 中指法譜與實際資料完全相符的佔比，與本研究中的指法譜 0/1 正確率 (Tablature 0/1 Accuracy) 是相同的定義。

接著如同上一章所提及，這份研究有做資料增強 (Data Augmentation) 的方法，所以上述兩項的分別是，前者是在原始資料中做訓練，後者是在增強過後的資料中做訓練，但兩者都在原始資料中做測試。

我們根據我們先前實驗的結果，#6 的參數量是我們所有訓練的模型中最好的，因此我們以這個架構為基準做一些調整，主要是 attention head 的數量，因為在前的數量 12 無法整除 162，在 PyTorch 無法執行，由於維度增加，我們選擇至最接近 12 且比 12 大的 18 作為 attention head 的數量。

我們先解釋該份研究 [1] 中的數值如何解讀，在此我們將其 Table 2. 數據表格重繪成 Table 6.8，這份研究設計了 data augmentation 的方法，四個大表格的分類是依據該結果中，augmentation 是否有應用在 training set、testing set，未使用 augmentation 則會加上 Guitar-only 的前綴修飾。接著是各個 column 的意義，依照該 token 中需要彈奏的音高數量做區分，因此有共有 6 種可能的值 (1~6)，接著是各個 row 的意義，如果該 token 中所有的指法皆相符，則歸類到 match，若部分相符，則歸類到 partial，若完全不相符，則歸類到 no match。至於 sum 則是將各個 row 或 column 作加總。

根據我們先前定義的指法譜 0/1 正確率，可以發現該數學意義與 (row, col) = (match, sum) 相同，也就是只計算指法完全相符的比率。由於我們的 test set 並未採用 augmentation，因此，我們不考慮 Augmented test 的情形，比較對象只有 Guitar-only test，其數值分別為 0.35 與 0.48，而我們的方法所訓練的模型則是 0.71，在數值上有明顯的領先。

不過這份研究僅使用了 dadaGP 資料集中 5% 的資料，如果該研究願意花更多

doi:10.6342/NTU202403031

時間或更好的設備，使他們能使用更多的資料來訓練模型，其表現結果仍應該能有所改善，所以我們並不能聲稱我們使用的模型架構一定有比較好的表現。但我們的研究不需要額外犧牲掉可用的資料，並且能在一週以內完成模型訓練，現階段的成果也確實比該研究中呈現的成果來得好。

	Guitar-only training - Guitar-only test							Augmented training - Guitar-only test						
num. p.:	1	2	3	4	5	6	sum	1	2	3	4	5	6	sum
no match	0.45	0.05	0.03	0.01	0.01	0.00	0.55	0.36	0.04	0.02	0.01	0.01	0.00	0.44
partial	0.00	0.03	0.05	0.01	0.00	0.00	0.09	0.00	0.03	0.03	0.01	0.01	0.00	0.07
match	0.25	0.05	0.03	0.01	0.00	0.01	0.35	0.34	0.06	0.05	0.01	0.01	0.01	0.48
sum	0.70	0.13	0.11	0.03	0.02	0.01	1.00	0.70	0.13	0.11	0.03	0.02	0.01	1.00
	Guitar-only training - Augmented test							Augmented training - Augmented test						
num. p.:	1	2	3	4	5	6	sum	1	2	3	4	5	6	sum
no match	0.16	0.16	0.02	0.03	0.03	0.05	0.45	0.12	0.15	0.02	0.03	0.03	0.05	0.40
partial	0.10	0.13	0.06	0.07	0.06	0.07	0.39	0.10	0.16	0.06	0.07	0.06	0.07	0.42
match	0.13	0.01	0.01	0.00	0.00	0.00	0.14	0.17	0.01	0.01	0.00	0.00	0.00	0.18
sum	0.29	0.31	0.09	0.10	0.09	0.12	1.00	0.29	0.31	0.09	0.10	0.09	0.12	1.00

Table 6.8: Cross-comparison results as a ratios over the total number of examined data instances when model was trained and tested with guitar-only and augmented data, when the transcription task incorporated 1 to 6 number of pitches.[1]

Dataset	Slot Accuracy	Tablature 0/1 Accuracy	Pitch Set Accuracy
Validation Set	0.8916	0.7150	0.8909
Testing Set	0.8947	0.7136	0.8919

Table 6.9: 指板位置上限改為 25 後，重新訓練的模型表現

6.4 輸出範例

我們以上一節的模型做為測試對象，輸入不存在於 dadaGP 資料集中的樂譜，測試其輸出結果。我們以 Right Here Waiting For You by Ulli Boegershausen[35] 的部分樂譜作為輸入，如 Figure 6.4所示，我們以原先的樂譜作為基底，以紅框標示模型應該要輸出但未輸出的部分，值得一提的是，在這份測試輸出中，並沒有發生錯位的情形，也就是未發生人類編排以某一弦彈奏特定音，模型卻選擇不同弦來彈奏相同音的情形，只有應彈奏而未彈奏的情形，所以我們只用到一種顏色的框來標示錯誤。

doi:10.6342/NTU202403031

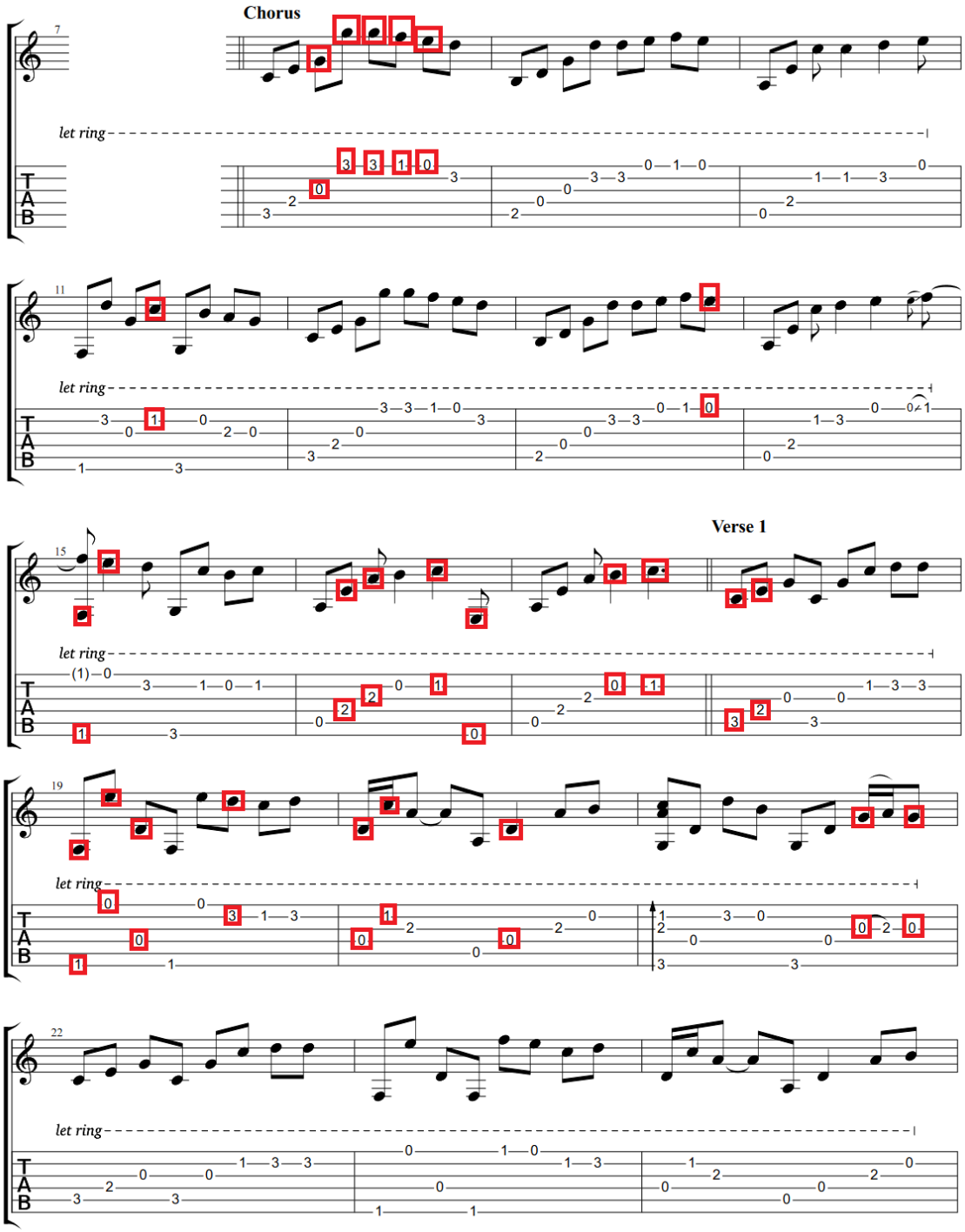

Figure 6.4: 原譜與輸出結果之比較，紅框處代表模型輸出為不彈奏的部分

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

第七章　結論

本章主要討論我們的貢獻與待研究議題。

7.1　貢獻

本節主要敘述我們認為本研究最重要的貢獻。

7.1.1　嵌入方法

我們初步驗證了4.3.1.1的嵌入方法有效性，也就是忽略節奏資訊的前提下，若單純是以樂譜轉譯為主的工作，本研究的嵌入方法就可以讓模型有不錯的表現。

7.1.2　找出適當的模型架構與參數量

我們比較了 attention head/self-attention layer 的數量以及 feed forward network 維度對於表現的影響，在我們的問題範圍內，實驗結果中目前得到最好的參數設定是 12 個 attention head，4 層 self-attention layer 以及 4096 維度的 feed forward network。

doi:10.6342/NTU202403031

7.2 待研究議題

本節主要討論本研究可能的改善方向，例如我們也許可以嘗試設計其他的嵌入方法，以及本研究問題定義排除了一些比較進階的技巧與設備條件，如果放寬這些條件，問題會變得複雜，但是也能增加應用的範圍。

7.2.1 嵌入方法改善以及考量更複雜的演奏技巧

我們的嵌入方法無法包含指法編排中的一些要素，例如，人的手大小有限，因此在僅考量左手按壓弦進行彈奏的前提下，理論上不太可能會出現有某個指法是其中一條弦按壓第 1 格，另一條弦按壓第 9 格，但如果只用左手按壓指板的前提不成立，透過點弦 [36] 的技巧可以解決前述的困境 (例如 Erik Mongrain 所表演的 Air Tap[37])。這個技巧是透過較用力、快速的方式按壓指板，彈奏特定的音高，是左右手都能夠進行的技巧。不過這些演奏技巧不一定會包含在指法譜，有時是依賴讀者自行查詢相關資料得知，所以如何找出這些資訊並納入輸入也是一項可以改進的課題。

7.2.2 納入不同的調音與 Capo 夾

不同的調音以及 Capo 夾的使用會讓指板位置與其彈奏出的對應音高有所變化，本研究的嵌入方法可能不再適用，必須另行設計，應該會增加嵌入維度，要得到一個好的方法會需要多方嘗試。

doi:10.6342/NTU202403031

7.2.3 更廣泛的指板位置

本研究設定的指板位置上限為 20，是考量到木吉他大多不會超過 20，若是要以電吉他為主，勢必得納入更高把位的資料，Table 7.1是 dadaGP 資料集中，不考慮弦的分別，各指板位置的彈奏次數，由此看來，在問題範圍中納入數值較高的 fret，其資料分佈會有長尾 (Long Tail) 問題，會是一項挑戰。

fret	count
0	11799101
1	3924286
2	11805394
3	9526698
4	5261572
5	8156690
6	3048766
7	6686176
8	2235282
9	3013625
10	1600870
11	638076
12	923266
13	233802
14	376983
15	218184
16	105681
17	162632
18	42660
19	70164
20	37098
21	19952
22	14557
23	2951
24	4912
25	1974
26	1355
27	647
28	566
29	622
30	575
31	1

Table 7.1: dadaGP 資料集中，不計弦別，發生在各指板位置的彈奏次數

doi:10.6342/NTU202403031

7.3 對於結果的評價

本研究的目的是使用找出一個方法，將一般樂譜轉換成吉他譜，最重要指標的有兩項，一是吉他譜要盡可能完整呈現需要彈奏的樂曲，二是指法編排要盡可能貼近人類的習慣，兩項數值同等重要必須兼顧。值得一提的是，兩者的關聯性較不對稱，也就是如果根本無法完整呈現樂曲，則可以得知指法編排肯定有錯，但就算完整呈現樂曲，指法編排也不一定正確。第一項我們使用 Pitch Set Accuracy 來評估，第二項使用 Slot Accuracy 來評估，兩者在最好的模型中都達到了接近 90%左右的數值，我們認為是相當好的結果。

doi:10.6342/NTU202403031

參考文獻

[1] M. Kaliakatsos-Papakostas, G. Bastas, D. Makris, D. Herremans, V. Katsouros, and P. Maragos, "A machine learning approach for midi to guitar tablature conversion," in Proceedings of the 19th Sound and Music Computing Conference, Jun. 2022.

[2] Keh Wei Lai, "Yiruma-river flows in you," 2015, [Online; accessed March 2, 2024]. [Online]. Available: https://fingerstyletw.blogspot.com/2015/03/yiruma-river-flow-in-you-sungha-jungtab.html

[3] 2020 MIDI Manufacturers Association, "Expanded status bytes list," 2020, [Online; accessed March 13, 2024]. [Online]. Available: https://midi.org/expanded-midi-1-0-messages-list

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[5] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," vol. 14, Jan. 2014, pp. 1532–1543.

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2017.

doi:10.6342/NTU202403031

[7] D. E. Rumelhart and J. L. McClelland, Learning Internal Representations by Error Propagation, 1987, pp. 318–362.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," CoRR, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," CoRR, vol. abs/2005.14165, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[12] A. Wiggins and Y. E. Kim, "Guitar tablature estimation with a convolutional neural network." in ISMIR, 2019, pp. 284–291.

[13] F. Cwitkowitz, J. Driedger, and Z. Duan, "A data-driven methodology for considering feasibility and pairwise likelihood in deep learning based guitar tablature transcription systems," arXiv preprint arXiv:2204.08094, 2022.

[14] E. Mistler, "Generating guitar tablatures with neural networks," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:30101141

doi:10.6342/NTU202403031

[15] G. Hori, H. Kameoka, and S. Sagayama, "Input-output hmm applied to automatic arrangement for guitars," Journal of Information Processing, vol. 21, pp. 264–271, Apr. 2013.

[16] G. Hori and S. Sagayama, "Hmm-based automatic arrangement for guitars with transposition and its implementation," in International Conference on Mathematics and Computing, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:17782959

[17] G. Hori and S. Sagayama, "Minimax viterbi algorithm for hmm-based guitar fingering decision," in International Society for Music Information Retrieval Conference, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:10478009

[18] Y.-H. Chen, Y.-H. Huang, W.-Y. Hsiao, and Y.-H. Yang, "Automatic composition of guitar tabs by transformers and groove modeling," 2020.

[19] P. Sarmento, A. Kumar, Y.-H. Chen, C. Carr, Z. Zukowski, and M. Barthet, "Gtr-ctrl: Instrument and genre conditioning for guitar-focused music generation with transformers," 2023.

[20] Bri Lundberg, "An easy guide to scientific pitch notation," 2022, [Online; accessed June 13, 2024]. [Online]. Available: https://www.musicandtheory.com/an-easy-guide-to-scientific-pitch-notation/

[21] Matthew Pinck, "How to read guitar tabs for beginners," 2022, [Online; accessed June 13, 2024]. [Online]. Available: https://www.benaturalmusic.live/how-to-read-guitar-tabs/

doi:10.6342/NTU202403031

[22] Ulli Boegershausen, “Right here waiting (by richard marx) - ulli boegershausen - solo guitar,” 2008, [Online; accessed June 14, 2024]. [Online]. Available: https://www.youtube.com/watch?v=HFWZk6jbVtE&ab_channel=Boegershausen

[23] Tommy Emmanuel, “Classical gas [mason williams] | tommy emmanuel,” 2011, [Online; accessed June 14, 2024]. [Online]. Available: https://www.youtube.com/watch?v=S33tWZqXhnk&ab_channel=TommyEmmanuel%2CCGP

[24] Jeff Owens, “Standard tuning: How eadgbe came to be,” n.d., [Online; accessed June 13, 2024]. [Online]. Available: https://www.fender.com/articles/setup/standard-tuning-how-eadgbe-came-to-be

[25] Jeff Owens, “What is a capo?” n.d., [Online; accessed June 13, 2024]. [Online]. Available: https://www.fender.com/articles/parts-and-accessories/what-is-a-capo

[26] Kim Perlak, “Guitar techniques: Hammer-ons, pull-offs, and slides,” 2022, [Online; accessed June 13, 2024]. [Online]. Available: https://online.berklee.edu/takenote/guitar-techniques-hammer-ons-pull-offs-and-slides/

[27] Nate Savage, “How to bend a note on guitar,” 2021, [Online; accessed June 13, 2024]. [Online]. Available: https://www.wikihow.com/Bend-a-Note-on-Guitar

[28] Simon Candy, “How to play percussive guitar,” n.d., [Online; accessed June 13, 2024]. [Online]. Available: https://acousticguitarlessonsonline.net/how-to-easily-add-percussive-techniques-to-your-acoustic-guitar-playing

[29] Patrick MacFarlane, “Harmonics,” n.d., [Online; accessed June 13, 2024]. [Online]. Available: https://www.guitarlessonworld.com/lessons/harmonics/

[30] P. Sarmento, A. Kumar, C. Carr, Z. Zukowski, M. Barthet, and Y.-H. Yang, “DadaGP:

doi:10.6342/NTU202403031

a Dataset of Tokenized GuitarPro Songs for Sequence Models," in Proceedings of the 22nd International Society for Music Information Retrieval Conference, 2021. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000076.pdf

[31] Sviatoslav Abakumov, "PyGuitarPro," 2014, [Online; accessed March 2, 2024]. [Online]. Available: https://pyguitarpro.readthedocs.io/en/stable/

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[33] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," vol. 14, Jan. 2014, pp. 1532–1543.

[34] PyTorch Contributors, "Transformer," 2023, [Online; accessed June 20, 2024]. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html

[35] class clef, "Right here waiting for you by ulli boegershausen (guitar tab)," 2012, [Online; accessed July 31, 2024]. [Online]. Available: https://www.classclef.com/pdf/Right%20Here%20Waiting%20For%20You%20by%20Ulli%20Boegershausen.pdf

[36] Lee Glynn, "Guitar tapping - a beginners guide to guitar tapping technique," 2020, [Online; accessed June 24, 2024]. [Online]. Available: https://www.pmtonline.co.uk/blog/2020/10/09/guitar-tapping-a-beginners-guide-to-guitar-tapping-technique/

[37] Erik Mongrain, "Erik mongrain "air tap" - music live show 2009, birmingham, uk (binaural sound for headphones)," 2009, [Online; accessed June 24, 2024]. [Online]. Available: https://www.youtube.com/watch?v=aqp8j4ghhgg

doi:10.6342/NTU202403031

doi:10.6342/NTU202403031

附錄 A — 模型訓練過程之損失數值走勢圖與驗證數值走勢圖

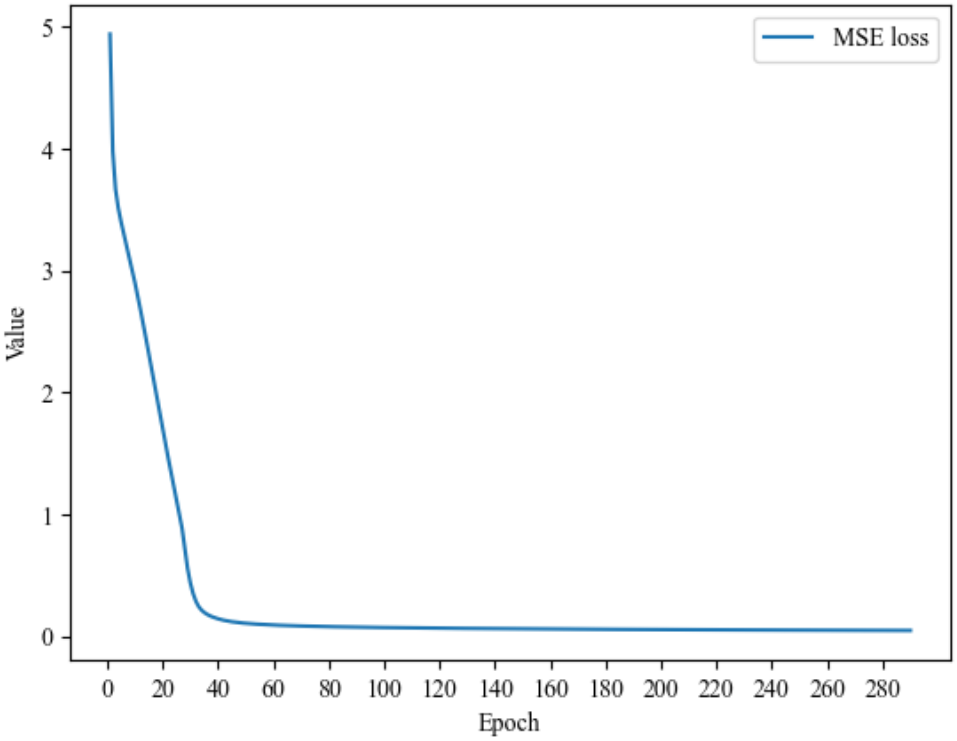

Figure A.1: Training Loss of Model#1

doi:10.6342/NTU202403031

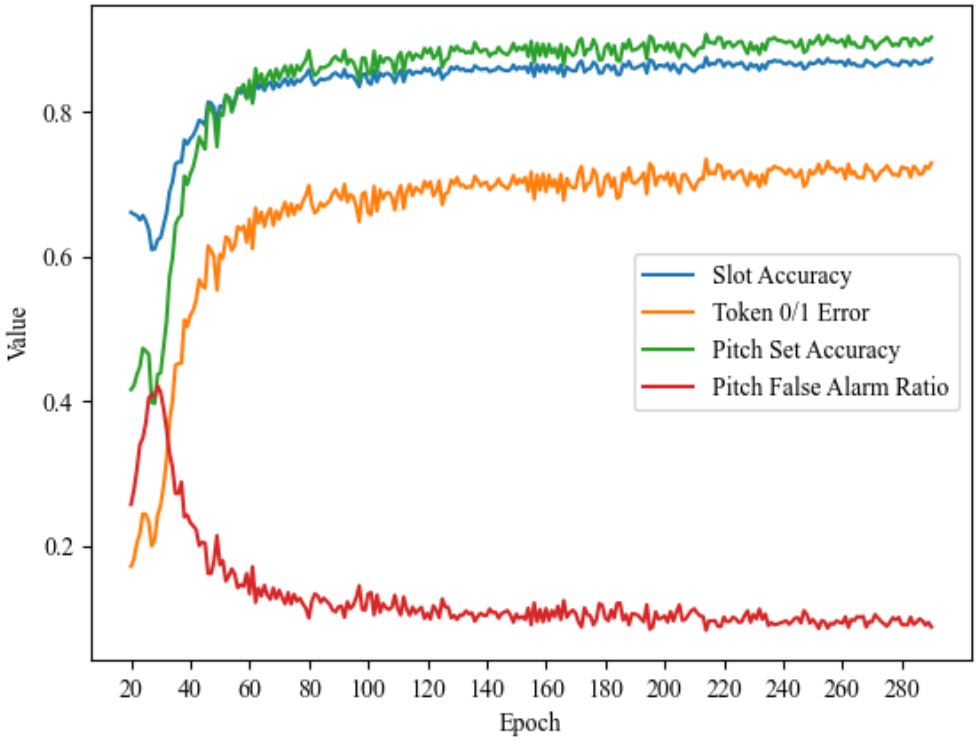

Figure A.2: Validation Progress of Model#1

doi:10.6342/NTU202403031

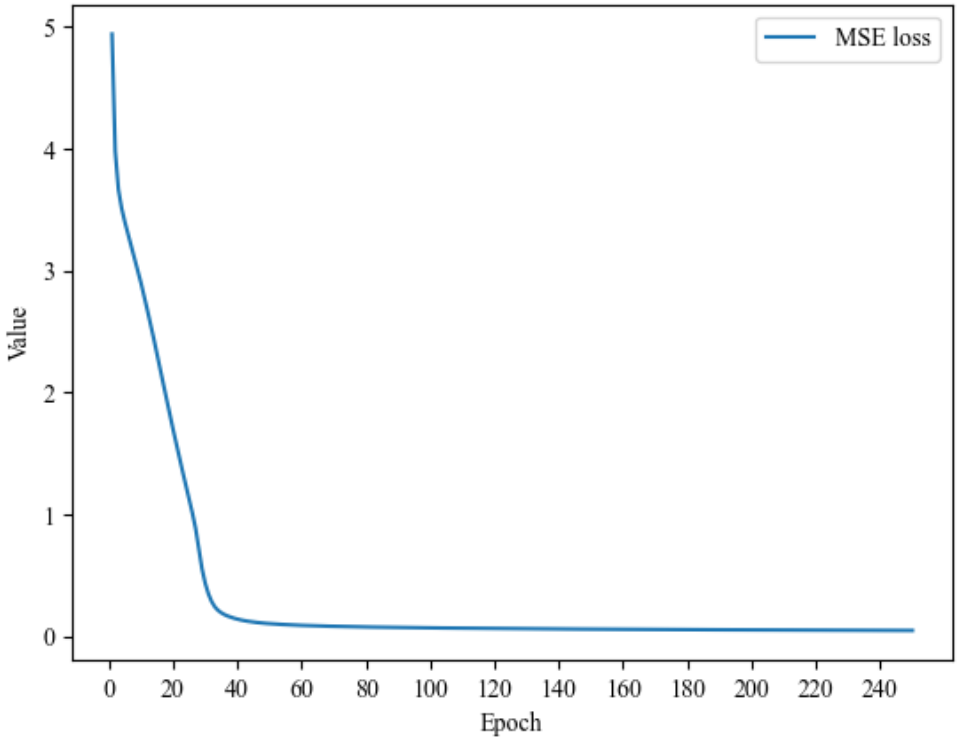

Figure A.3: Training Loss of Model#2

doi:10.6342/NTU202403031

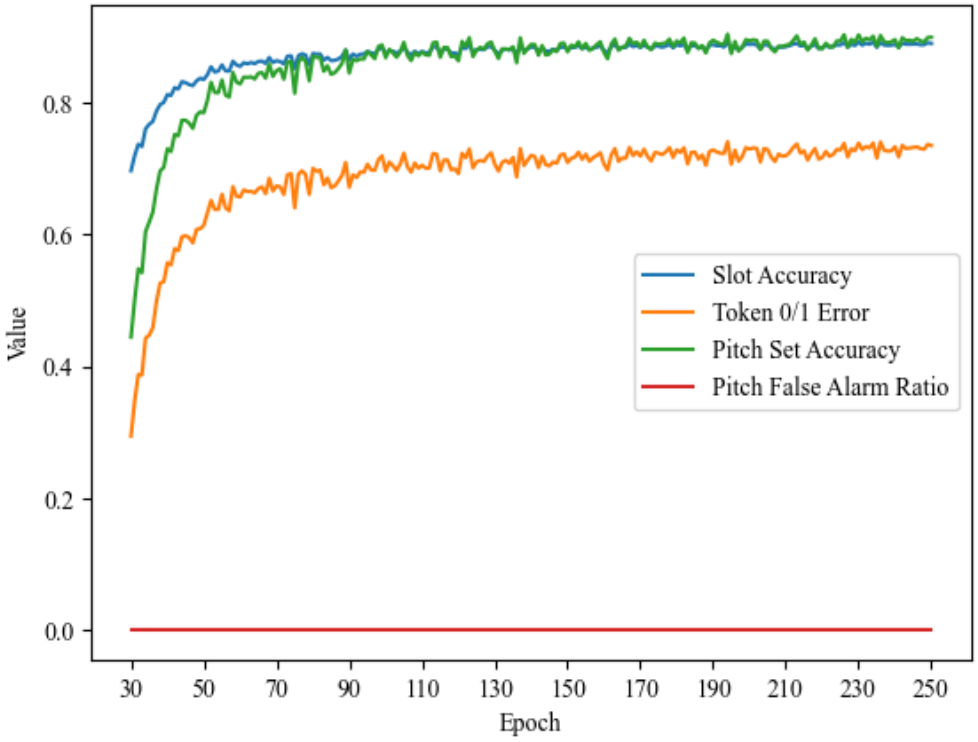

Figure A.4: Validation Progress of Model#2

doi:10.6342/NTU202403031

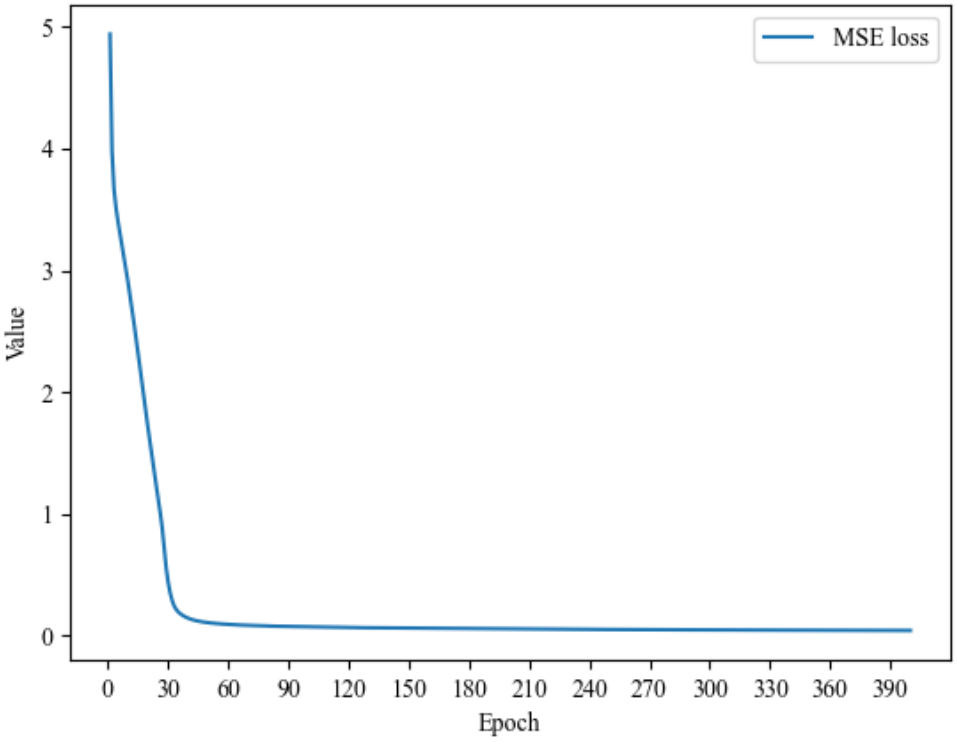

Figure A.5: Training Loss of Model#3

doi:10.6342/NTU202403031

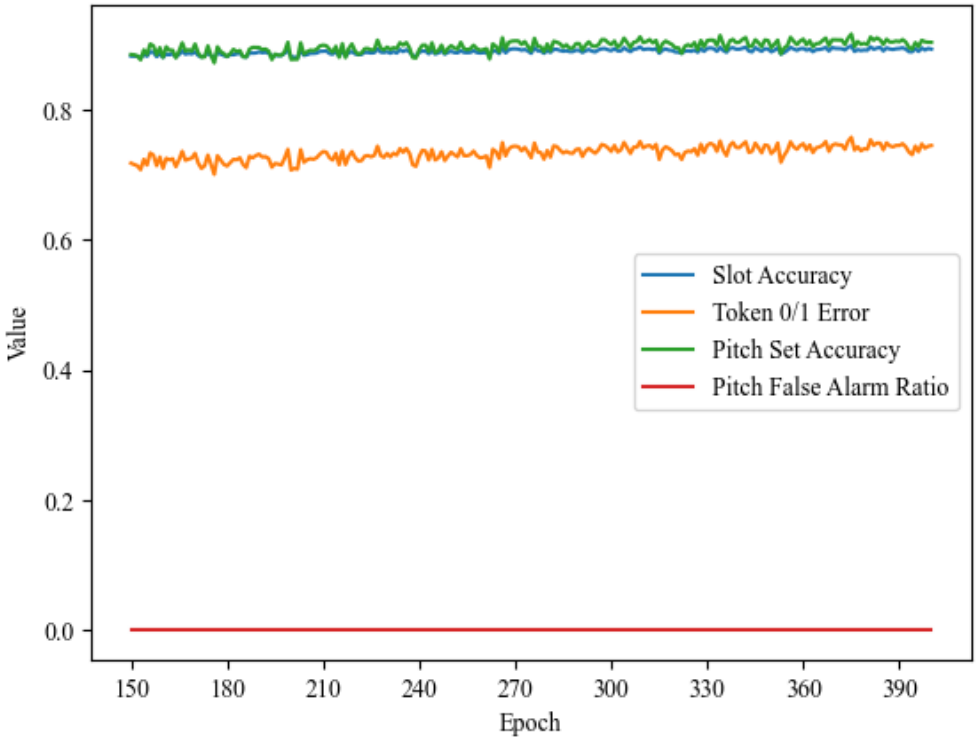

Figure A.6: Validation Progress of Model#3

doi:10.6342/NTU202403031

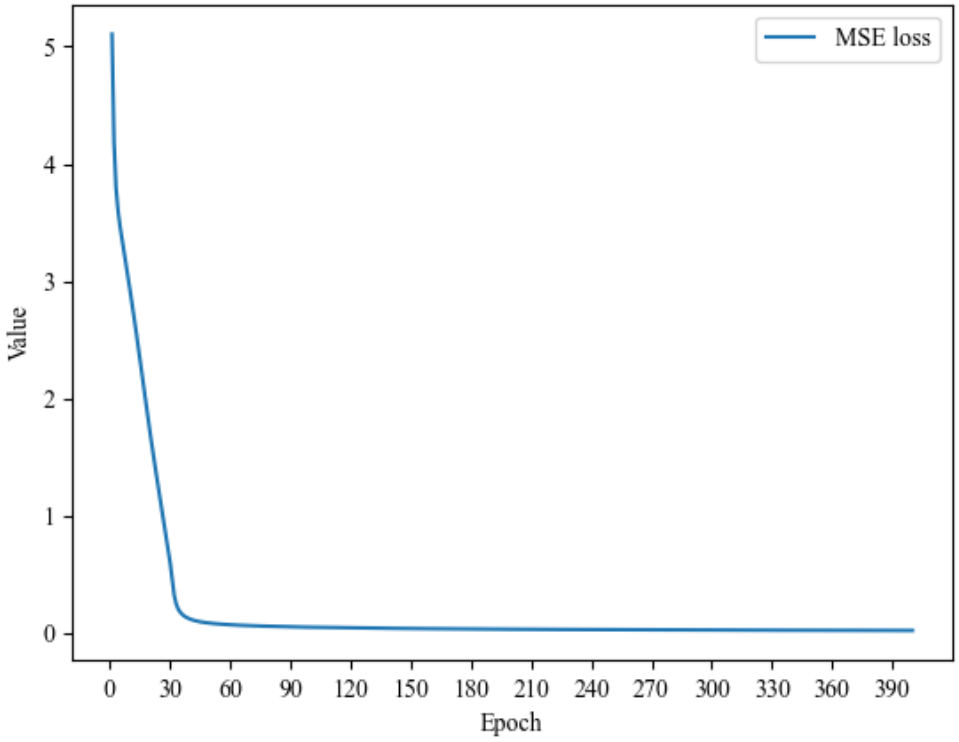

Figure A.7: Training Loss of Model#4

doi:10.6342/NTU202403031

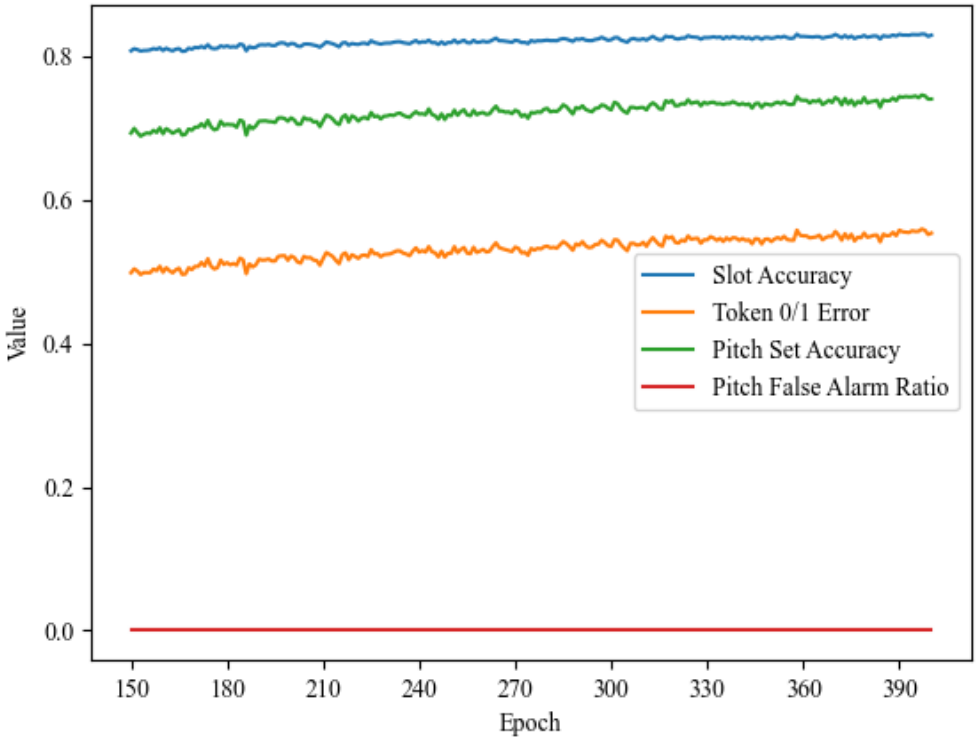

Figure A.8: Validation Progress of Model#4

doi:10.6342/NTU202403031

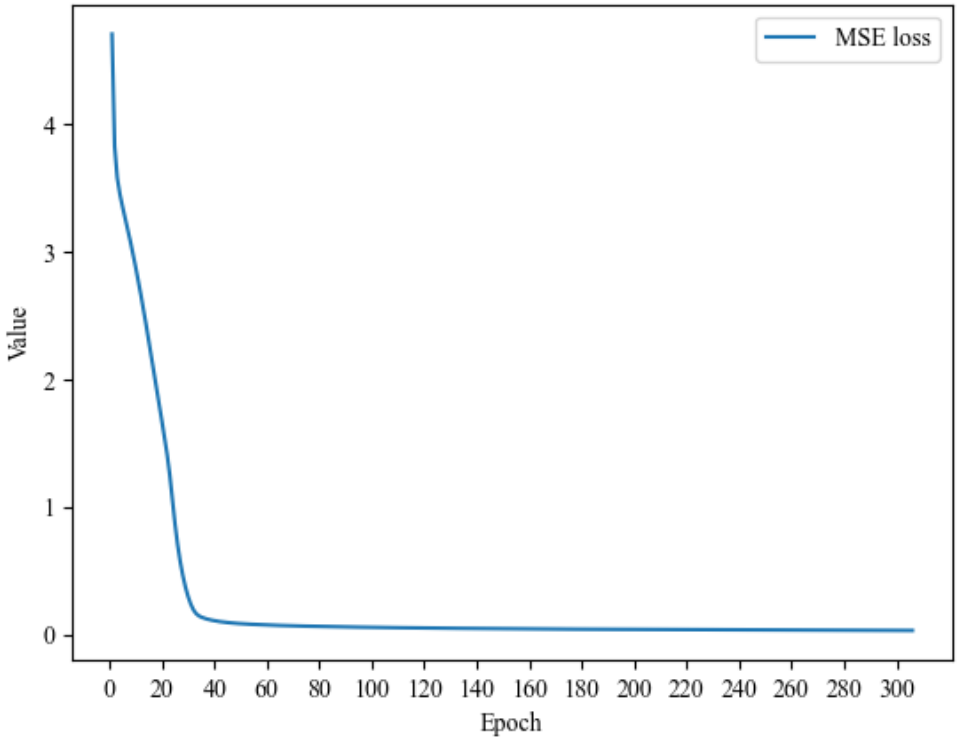

Figure A.9: Training Loss of Model#5

doi:10.6342/NTU202403031

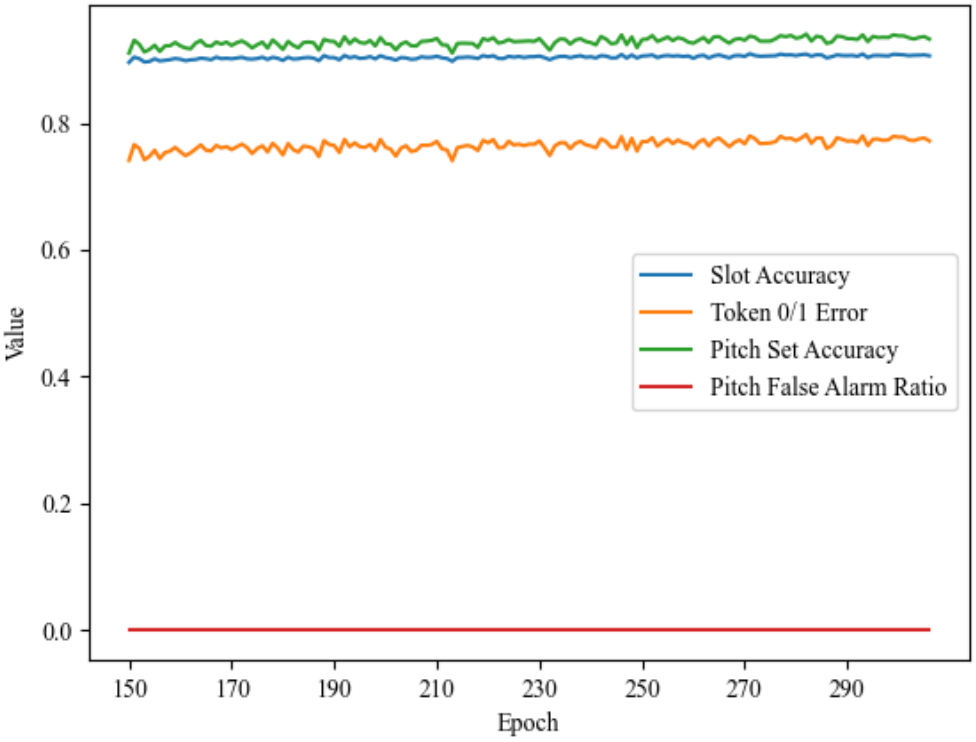

Figure A.10: Validation Progress of Model#5

doi:10.6342/NTU202403031

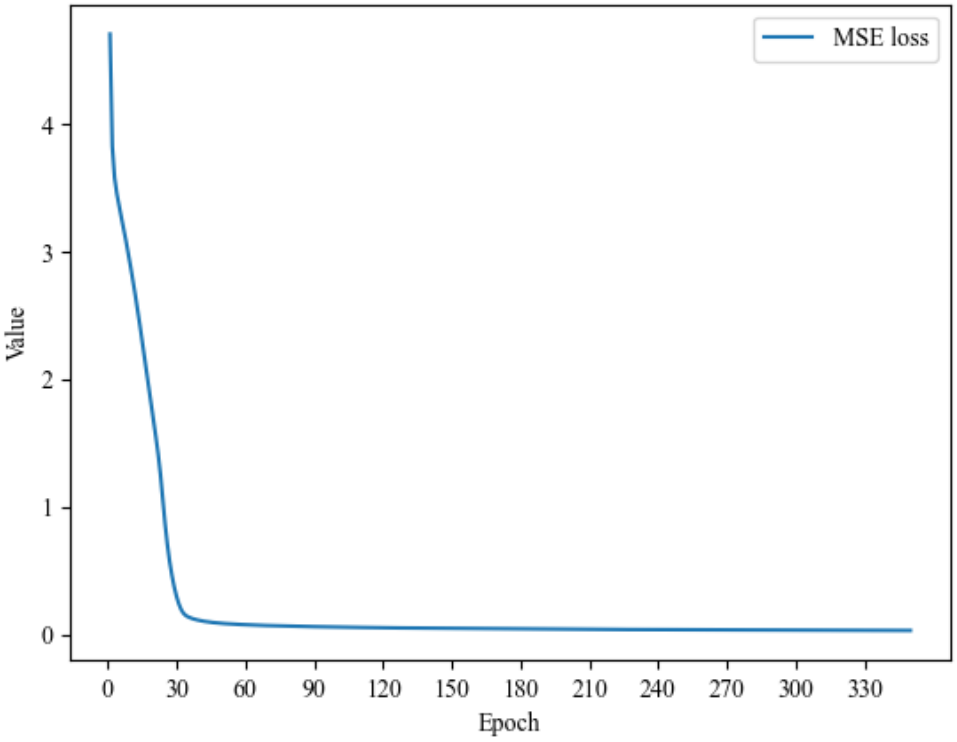

Figure A.11: Training Loss of Model#6

doi:10.6342/NTU202403031

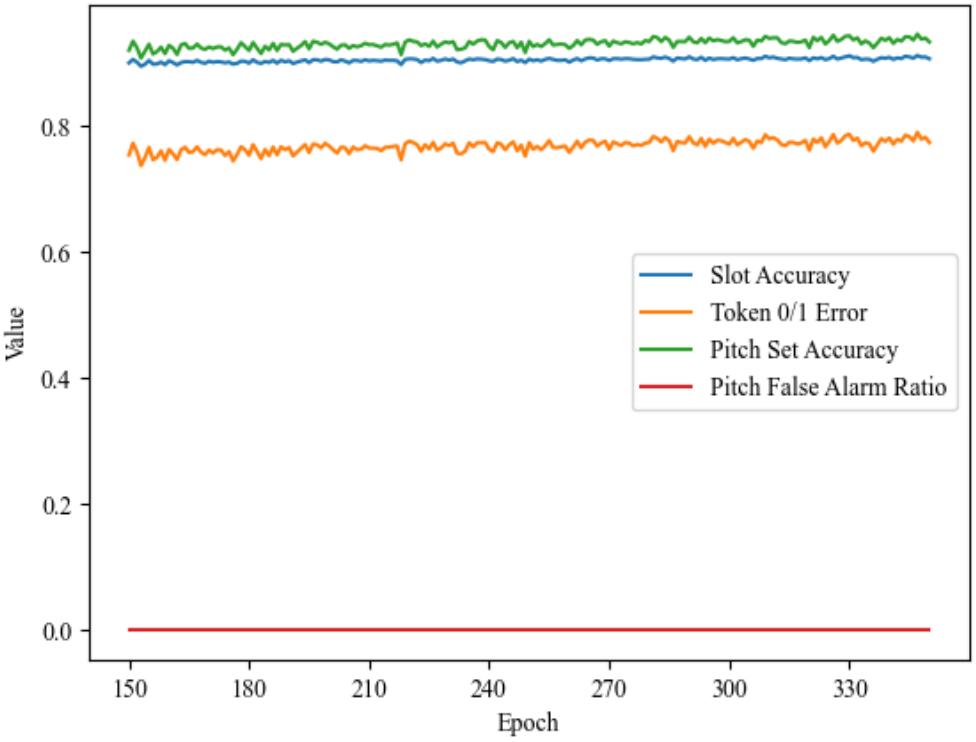

Figure A.12: Validation Progress of Model#6

doi:10.6342/NTU202403031

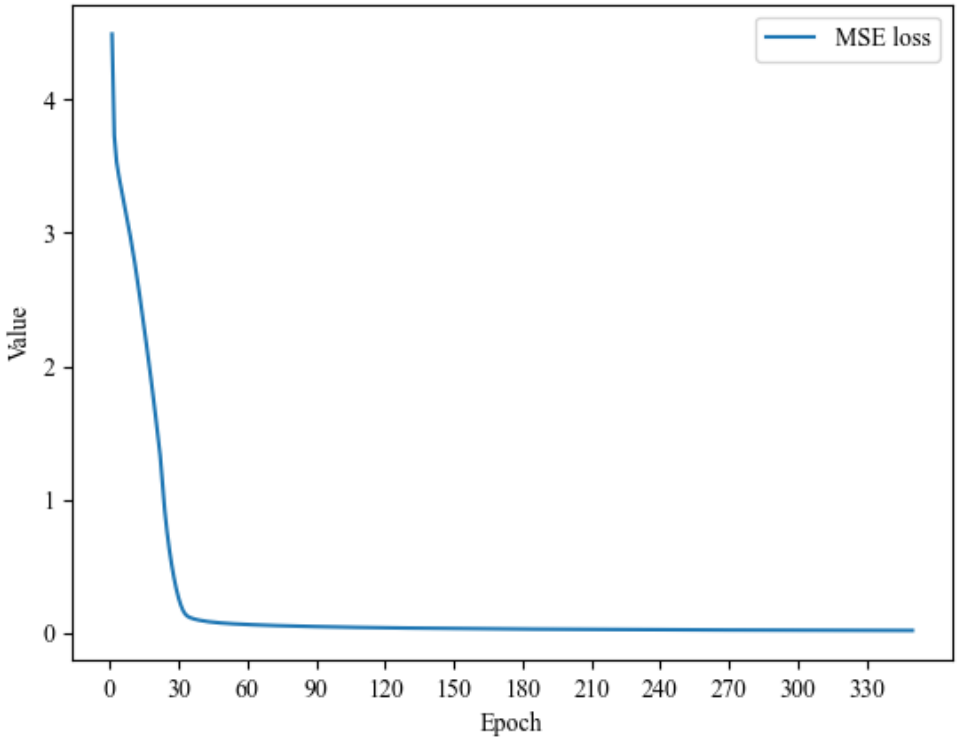

Figure A.13: Training Loss of Model#7

doi:10.6342/NTU202403031

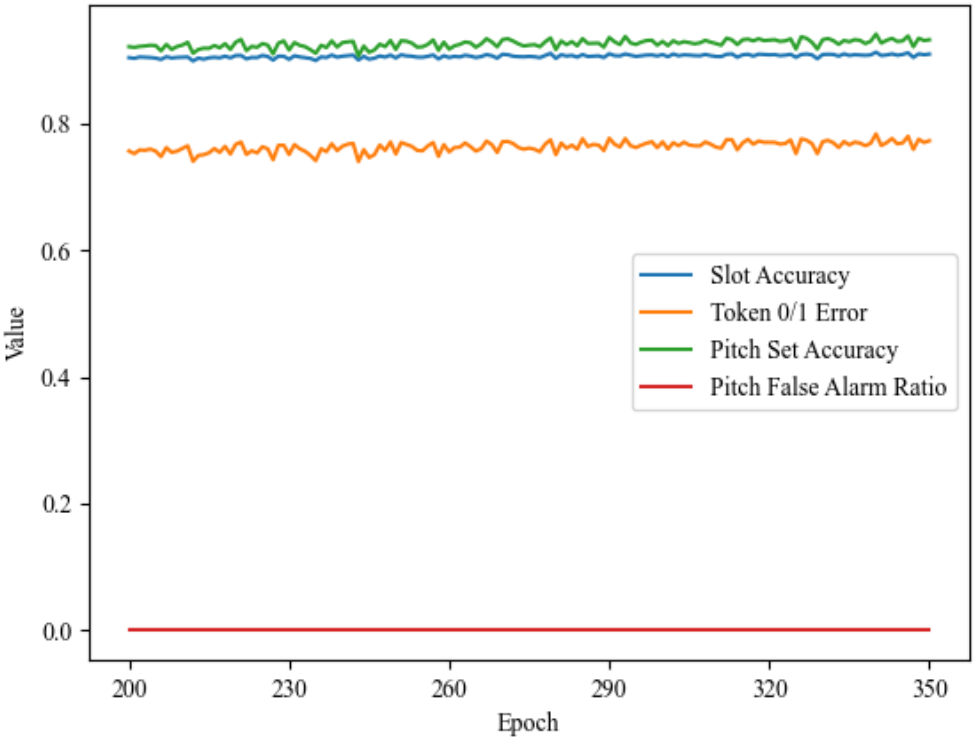

Figure A.14: Validation Progress of Model#7

doi:10.6342/NTU202403031

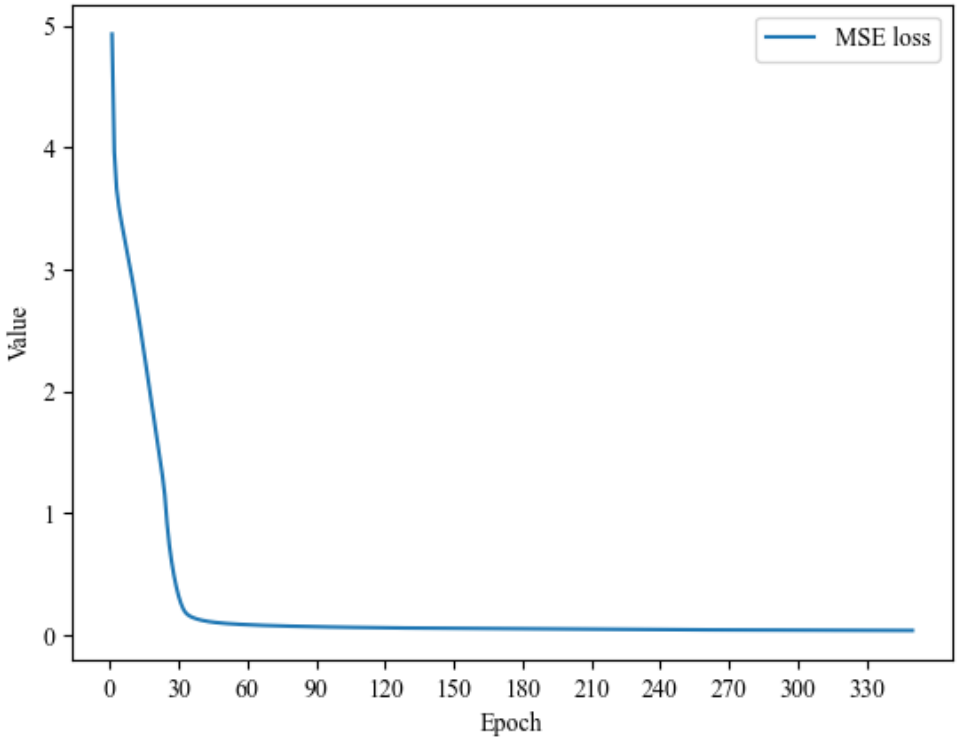

Figure A.15: Training Loss of Model#8

doi:10.6342/NTU202403031

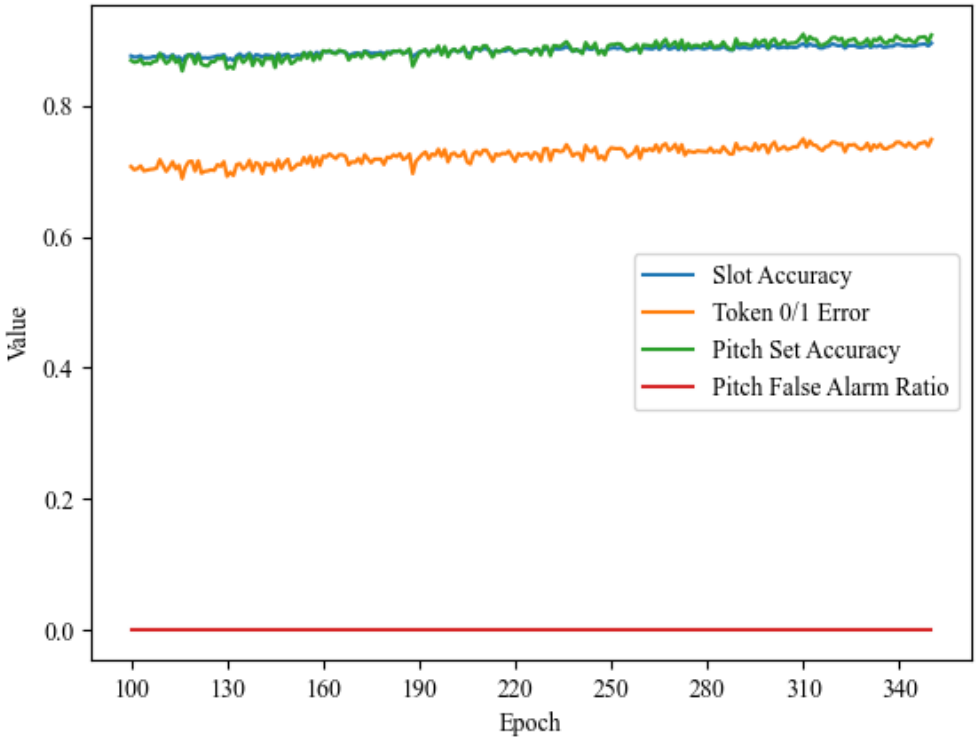

Figure A.16: Validation Progress of Model#8

doi:10.6342/NTU202403031

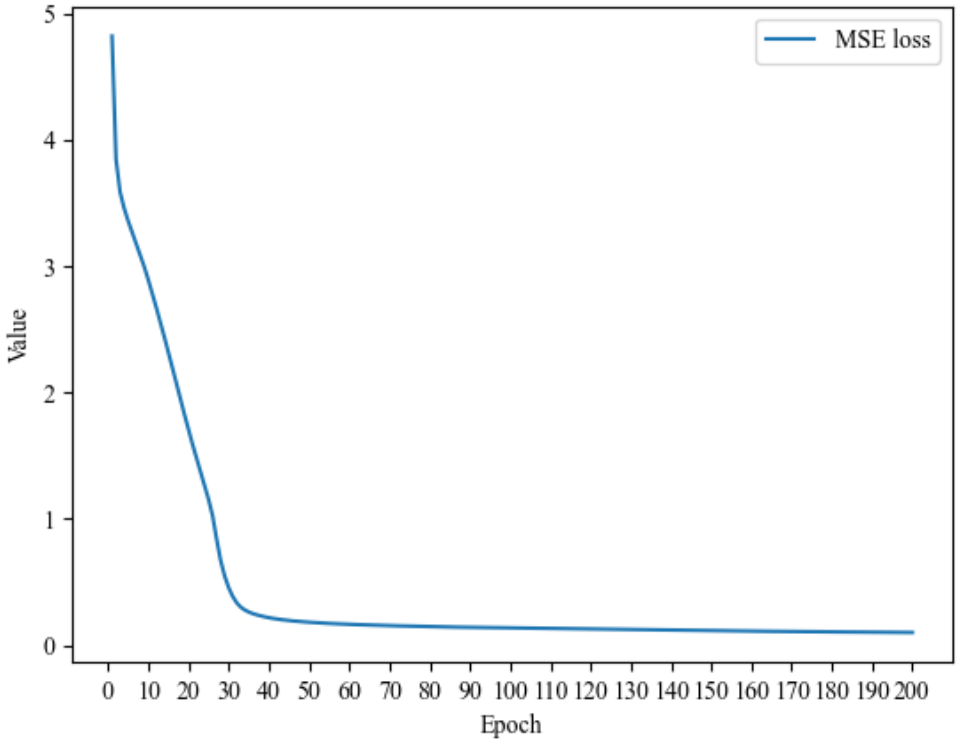

Figure A.17: Training Loss of Model#9

doi:10.6342/NTU202403031

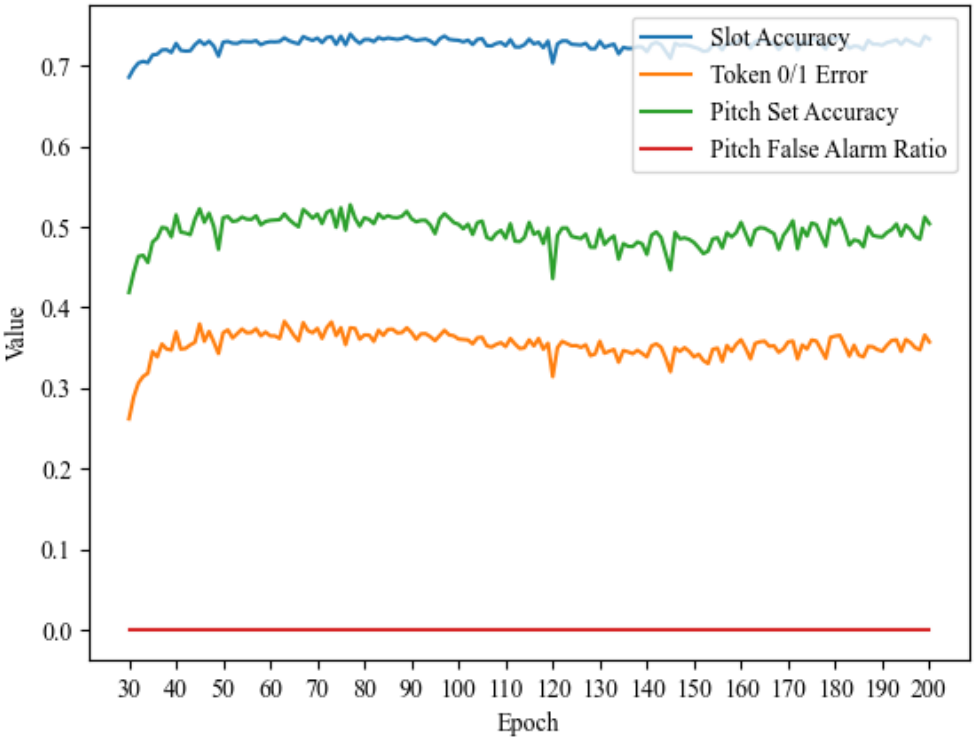

Figure A.18: Validation Progress of Model#9

doi:10.6342/NTU202403031

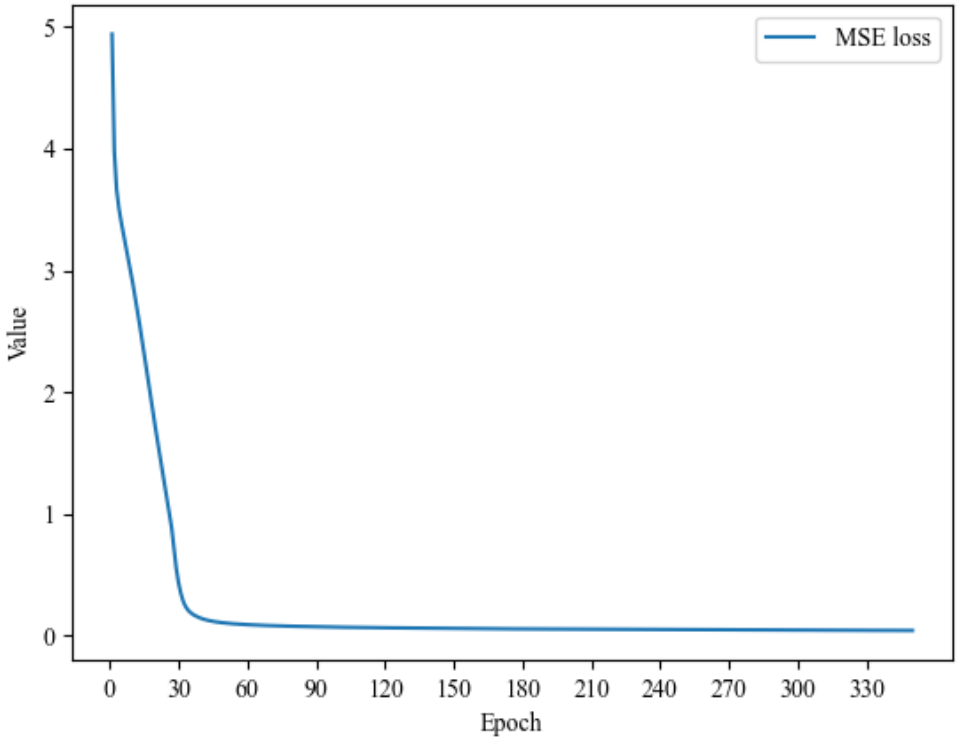

Figure A.19: Training Loss of Model#10

doi:10.6342/NTU202403031

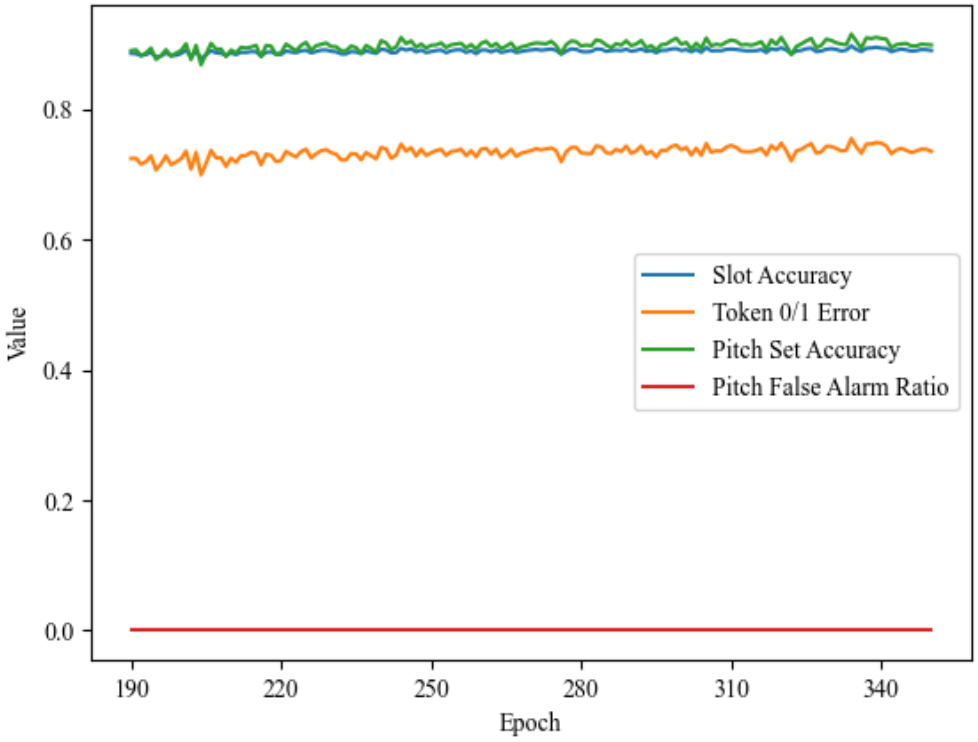

Figure A.20: Validation Progress of Model#10

doi:10.6342/NTU202403031

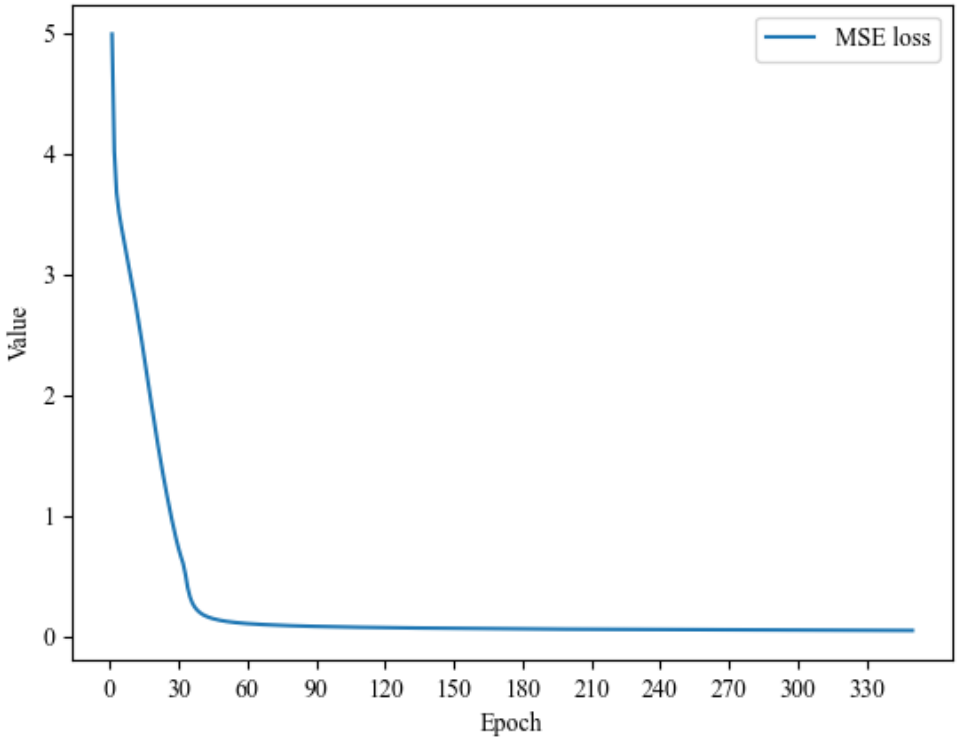

Figure A.21: Training Loss of Model#11

doi:10.6342/NTU202403031

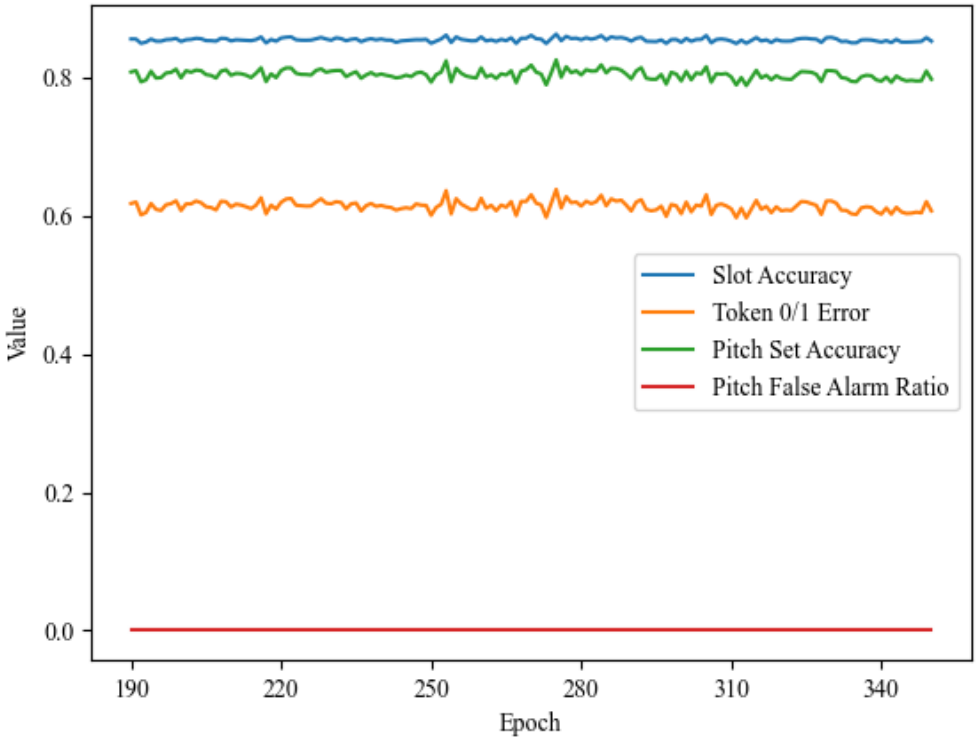

Figure A.22: Validation Progress of Model#11

doi:10.6342/NTU202403031

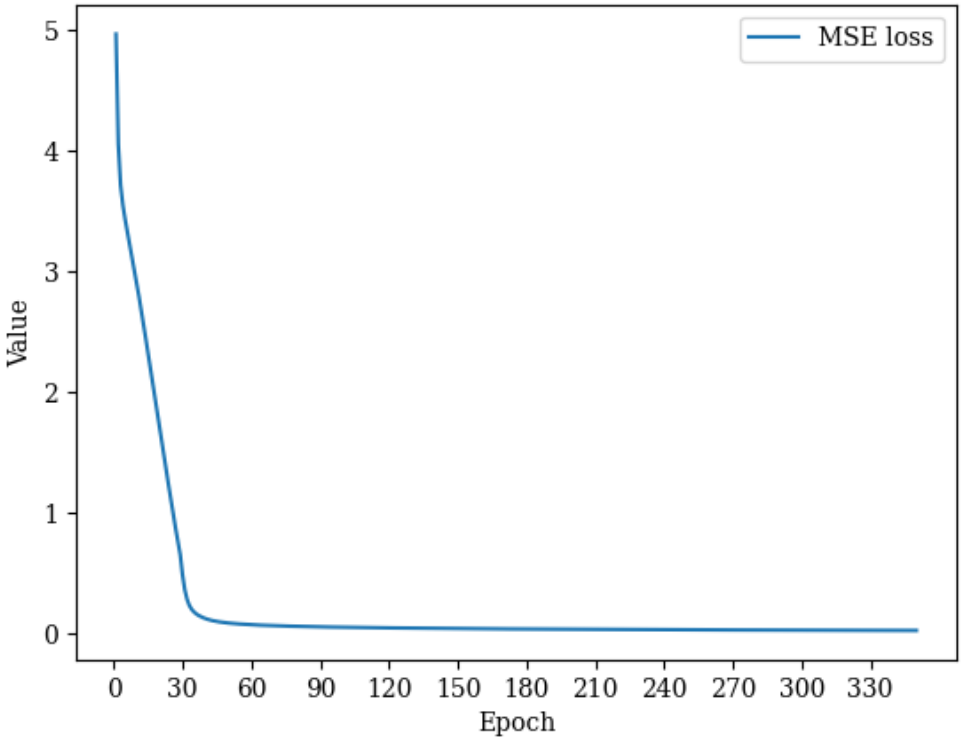

Figure A.23: Training Loss of Model#12

doi:10.6342/NTU202403031

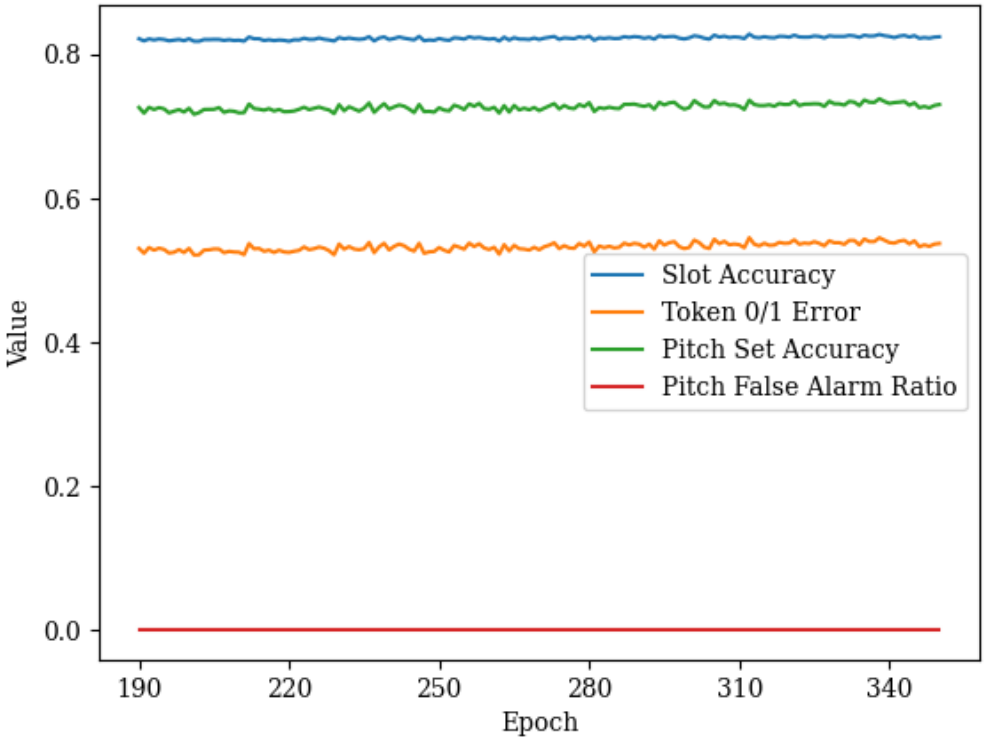

Figure A.24: Validation Progress of Model#12

doi:10.6342/NTU202403031