Otamatone 實作

第一屆毫米波雷達 AI 創意競賽－天上太陽紅通通

陳奕羶
資訊工程學系
國立暨南國際大學
南投, 臺灣
s110321015@ncnu.edu.tw

張簡雲翔
資訊工程學系
國立暨南國際大學
南投, 臺灣
s110321018@ncnu.edu.tw

劉德權
資訊工程學系
國立暨南國際大學
南投, 臺灣
s110321064@ncnu.edu.tw

王駿彥
資訊工程學系
國立暨南國際大學
南投, 臺灣
s110321069@ncnu.edu.tw

Abstract

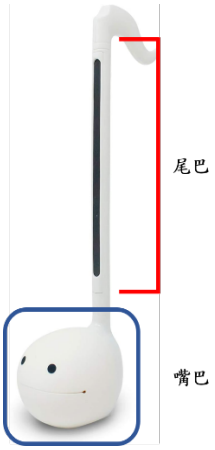
Otamatone 是一種外型類似八分音符的電子樂器，其操作方式與弦樂器類似且更為簡單，而小巧的體積也提供了良好的便攜性。然而 Otamatone 也有一些缺點，由於其價格較高，一般家庭和學生較難負擔，而音色也較為單一不能隨意更改。為此，本次實驗希望透過毫米波雷達加上影像辨識的技術，在省略滑音功能的情況下，設計出價格低廉且音色多元的簡化版 Otamatone。最終選用 R(2+1) D CNN 架構作為最終的模型架構，並在 test set 上得到 0.61 的 accuracy，雖然實際部屬後的結果表現的不是很理想，但仍舊對於毫米波的未來提出了一個可能發展的趨勢。

Keywords ─ 毫米波雷達、Video Classification

I. INTRODUCTION

Otamatone 是一個電子樂器，外型類似八分音符，結構分為上下兩個部分：尾巴與嘴巴（圖一）。尾巴的部分對應符桿，負責控制音的高低，控制方法與目前的弦樂器類似，演奏者將手指依靠在桿子上，並透過手指所在的位置決定當前要發出的音高是什麼，越靠近桿子頂部，音高越低，反之則越高；嘴巴則對應符頭的部分，演奏者透過按壓嘴巴即可發出聲音。相比於一般常見的弦樂器，Otamatone 的操作相對簡單，可以將其類比成只有一根弦的弦樂器，因此，演奏者只需要將注意力集中在同一根弦上，不需要注意每根手指的按壓情況以及要彈撥對應的弦。此外，吉他、貝斯等弦樂器，演奏者在練習或是演奏時都需要忍受手指按壓和撥動弦所帶來的疼痛感，而 Otamatone 的設計結構可以讓演奏者免去這樣的缺點，可以讓演奏者隨心所欲的演奏和練習。雖然 Otamatone 有操作簡單以及便於攜帶等特點，但是仍有一些缺點：價格方面，市面上的售價幾乎都超過一千元以上，這對於一般的家庭以及學生仍是不小的支出；在音色方面，Otamatone 也只有一種音色供演奏者使用，不能隨意的更改。

綜觀上述，Otamatone 這個樂器十分的有趣，但因為價格偏高且功能上有所限制，所以本次實驗想要藉由毫米波套件設計出簡化版的 Otamatone：在省略掉滑音功能，留下指壓發音的功能外，增加其音色的多樣性。



圖一：Otamatone 架構

II. BACKGROUND

A. K60168A Dongle 套件

K60168A 晶片是由開酷科技自行研發，主要包含了 60GHz 的毫米波雷達及 AI 加速器等關鍵先進技術整合成一個系統單晶片(SoC)，此外還有 1 個發射天線和 3 個接收天線負責收發訊號，偵測範圍內辨識出物體的方位、距離、速度；此外，開酷科技也提供了軟體及原始碼用以收集手勢影像及擷取即時資料。本次實驗將會使用 K60168A 晶片以及官方軟體收集資料並建構整體系統架構。

B. Video Classification 模型

1. ConvLSTM

在 2015 年被提出[1]，其在 LSTM 的基礎上進行了改良，同時結合了 CNN 架構以及 LSTM 架構的特性，將 LSTM 中部分的矩陣運算，改成了 CNN 的運算方式。這樣的改動使其可以處理同時具有圖片以及時序特性的資料，例如影片等。

2. R(2+1)D CNN

在 2017 年由 Meta 所提出來的模型[2]，一般 CNN 要處理影片會採用 Conv3D 的方式來處理，而 R(2+1)D CNN 使用類似 Depthwise Separable Convolution[3]的方式將原先 Conv3D 的作法拆解成 Spatio 部分跟 Temporal 部分((2+1D) Convolution)，以減少運算量並降低 overfitting 的機會。此外，也引入了 skip connection 的概念來避免 Degradation problem[4]。

3. ViViT

在 2021 年由 Google 提出的一個模型[5]，基於 Vision Transfomer（ViT）模型進行了改良。ViT 只用於 2D 圖片的分類，由於影像有時間的維度，故 ViViT 在做影像的辨識也導入了時間的維度。在論文中提出了四種不一樣的架構（圖二）其中第一種是單純的 Transformer，後面三種是在建構時間與空間的 Transformer。

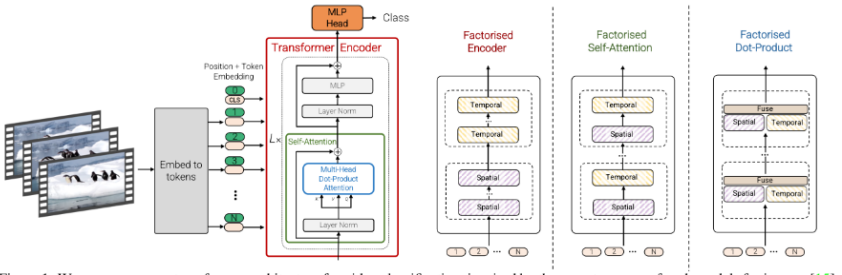


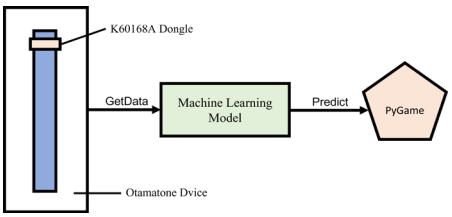
Figure 1: We propose a pure-transformer architecture for video classification, inspired by the recent success of such models for images [15]. To effectively process a large number of spatio-temporal tokens, we develop several model variants which factorise different components of the transformer encoder over the spatial- and temporal-dimensions. As shown on the right, these factorisations correspond to different attention patterns over space and time.

圖二： ViViT 的 4 種架構

本次實作中預計會使用 ConvLSTM、R(2+1)D CNN、CNN 三個影像辨識的模型，由於 ViViT 所需要的算力與時間成本較高，因此在經過考量後決定不使用該架構。

III. SYSTEM STRUCTURE

此次實驗將會將 K60168A Dongle 固定在一根長度約 15~20 公分的桿子上，藉此來模擬 Otamatone 的結構。當 K60168A Dongle 接收到訊息後，會連接到電腦並輸入至模型進行預測，模型的預測結果會輸入至 PyGame 並調用 Windows 中內建的合成器撥放聲音。原先是採用 Sonic Pi 來撥放對應的音樂，不過其內建的合成器聲音品質不太好，而 PyGame 除了聲音較好外，且可以從電腦上選擇不同的樂器聲音，例如：鋼琴、吉他、小提琴等，故最後決定改用 PyGame。

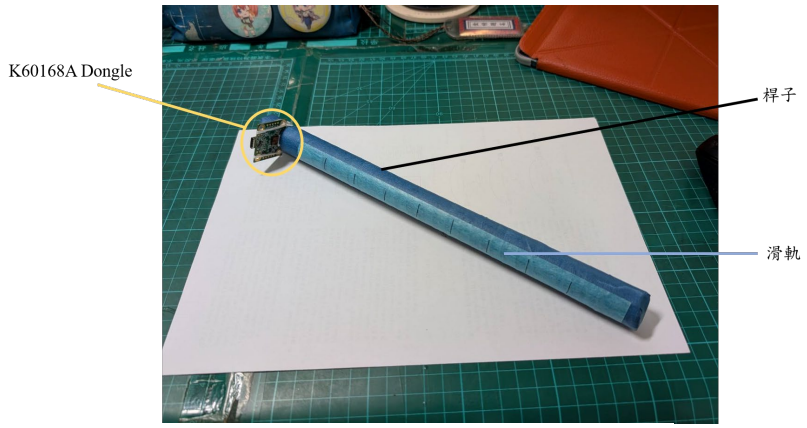


圖三：系統架構圖

IV. IMPLEMENTATION AND DIFFICULT

A. 裝置實作

在一根長度約為 30 公分的桿子，距離頂端約 5 公分處鋸出一個開口，將 K60168A Dongle 固定該開口處並用棉繩將其綁在桿子上，後面的 25 公分纏上膠帶並均分成 9 等分且用黑筆標記位置，做為滑軌使用（圖四）。



圖四: 裝置成品圖

B. 資料收集

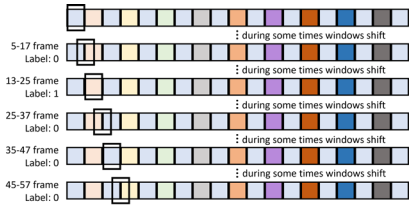
藉由開酷科技提供的 Collect_RDI 軟體裡面的 pre-define 的 mode，設定每次動作對應到的 frame 以及其間隔，其中每個動作固定為 12 frames，且間隔數個 frames，由於硬體效能的限制導致資料前處理過久，所以總共收集的 frame 數量為 190 個，且為經過 Fourier Transform 的兩張 32*32 的圖片。

C. 資料分割

本次實驗合計收集了 307 筆的資料，並作為模型的資料集。每筆資料裡面都有 10 種 label，並將其切成三份作為 training set、validation set、test set，其中 192 筆為 training set，占全部的 64%；49 筆為 validation set，占全部的 16%；62 筆為 test set，占全部的 20%。

D. 資料預處理

原始資料共有 190 個 frames 並有多個 label 分別如下：0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 0，label 0 表示沒有動作；label 1~7 由低到高分別為 Do Re Mi Fa So La Si；label 8 代表聲音上調；label 9 代表聲音下調。為了後續訓練方便，所以需要把各個 label 的資料從單一資料分割出來，此次使用移動 window 的方式對資料進行掃描與切割，window size 設為 12，只有 window 中所有 label 皆為一樣，才標記為該 label，其他的則視為 label 0。此外，為避免資料產生極大的偏態，所以訓練時只有隨機選取與其他 label 一樣量的 label 0（圖五）。



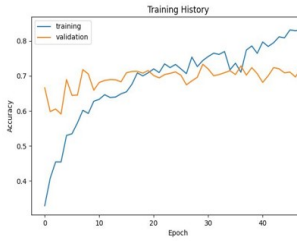
圖五: window 掃描示意

E. 模型架構

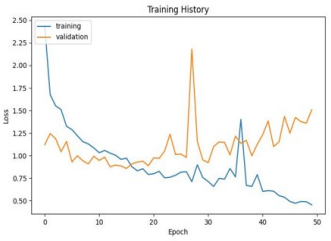
本次實作採用了 9 種不同的模型架構，epoch 都設定為 50 ，其中又可以細分成 2 種 CNN 模型架構、4 種 ConvLSTM 模型架構以及 3 種的 R(2+1)D CNN 模型架構。

1. CNN Version 1

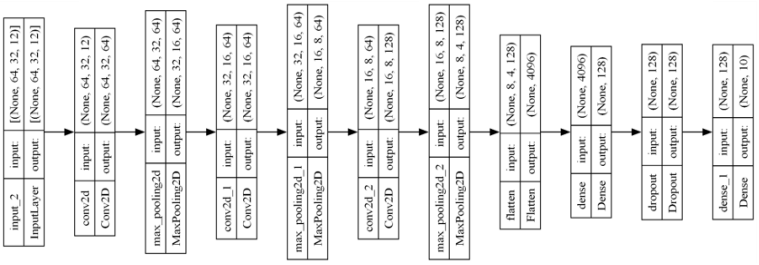
由於影片的資料可以當作一張擁有很多 channel 的圖片，且經由小組成員討論，認為手勢的動作並不複雜，為此，最後決定先使用一般的 CNN 模型來嘗試訓練（圖六(c)）。在資料的處理上，除了先前的資料預處理，還會將每個 frame 的兩張 32*32 的圖片合併為 64*32 的圖片，並且將每個 frame 當成一個 channel 疊成一張 shape 為 12*64*32 的圖片，最後再用 Conv2D 掃描。總參數量為 969,098，在 validation set 上得到 accuracy 為 0.7097，表現還算能接受，不過模型在最後訓練的部分有些出現 overfitting 的情況（圖六(a)(b)），所以還需要再做些微調。



(a) Accuracy



(b) Loss



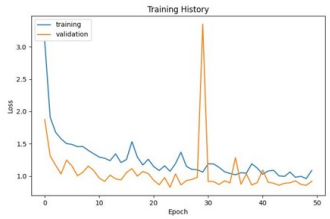
(c) 模型架構圖

圖六：CNN Version 1 Learning Curve(a)(b)與模型架

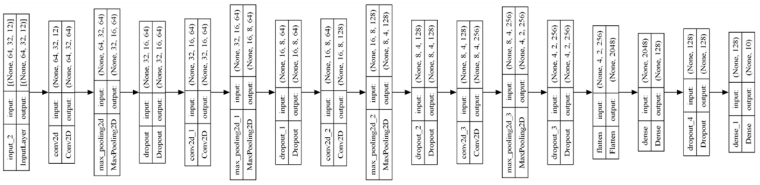
2. CNN Version 2

相較於 CNN Version 1，透過在每個 pooling layer 後面加上一個 0.3 的 dropout layer，並在兩層 fully connected layer 中間加上一個 dropout rate 為 0.5 的 dropout layer（圖七(c)），來改進 CNN Version1 的 overfitting 的問題。總參數量為 1,002,122 訓練的結果在 validation set 上得到 accuracy 為 0.6255，雖然有成功抑制了 overfitting 的發生，不過對比前一版效能退步的很明顯（圖七(a)(b)）。由於 CNN 本身是 high varience 的[6]，因此要提升表現，除了增加參數量，還需要增加資料量[7]或者是加入 dropout layer 來抑制 overfitting。經由小組成員的討論，除非繼續增加資料量，否則模型的表現可能已經到極限了，礙於時間問題，因此後續轉而使用其他的模型架構。

(a) Accuracy



(b) Loss

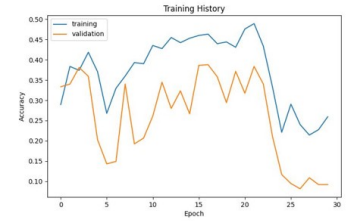


(c) 模型架構圖

圖七：CNN Version 2 Learning Curve(a)(b)與模型架構(c)

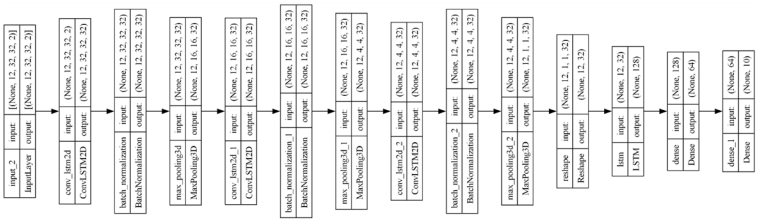
3. ConvLSTM Version 1

由於影片是一種具有 time step 概念的資料，考慮到 CNN+LSTM 的組合無法同時捕捉圖片以及時序上的資訊，因此採用可以捕捉圖片、時序資訊且參數量較小的 ConvLSTM 架構[1]。ConvLSTM Version 1 的總參數量為 278,730（圖八(c)），在 validation set 上得到 accuracy 為 0.0922。最後的表現除了不理想外，在訓練過程中也出現了 gradient explosion（accuracy 在後半部分突然驟降、loss 急速上升）（圖八(a)(b)），推測有可能是本身模型的效能表現就是如此、加了 LSTM layer、資料集沒有完整的打亂又或是 optimizer 選擇的問題。



(a) Accuracy

(b) Loss



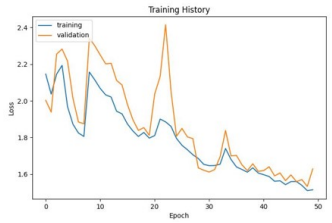
(c) 模型架構圖

圖八：ConvLSTM Version 1 Learning Curve(a)(b)與模型架構(c)

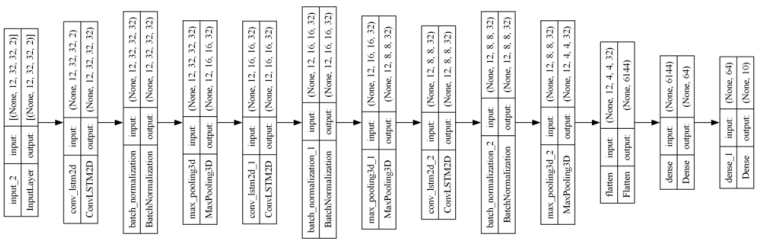
4. ConvLSTM Version 2

在前一版模型中，推測可能是模型本身的問題、資料集沒有完整打亂或是 optimizer 的問題。經由小組成員的討論，決定先測試是不是模型本身和 LSTM layer 的問題，為此，將原先的 kernel size 拉大並且去除 LSTM layer，最終的總參數量來到 782,026（圖九(c)）。最後在 validation set 上得到 accuracy 為 0.3221，相較於 ConvLSTM Version 1 的表現提升了不少，但是模型的表現仍舊是不太理想（圖九(a)(b)）。

(a) Accuracy



(b) Loss



(c) 模型架構圖

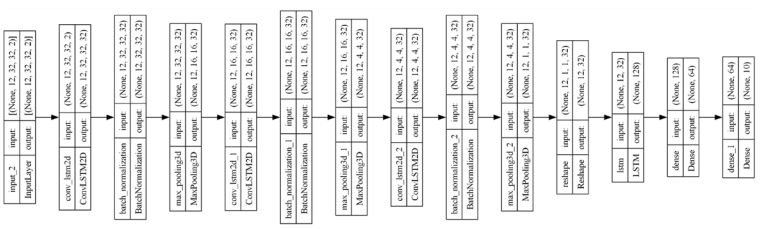
圖九：ConvLSTM Version 2 Learning Curve(a)(b)與模型架構(c)

5. ConvLSTM Version 3

在 ConvLSTM Version 2 中，雖然增加模型參數量以及去除了 LSTM layer，但是其效果並沒有到很好，經由討論，猜測可能是資料沒有打亂的緣故。為此，在訓練前先將原本的資料全部按照 label 排序好，並透過 Tensorflow 的 shuffle 功能來打亂資料順序，此外因為沒有對架構進行調整，所以參數量仍然維持在 782,026（圖十(c)）。最後在 validation set 上得到 accuracy 為 0.214，表現對比 ConvLSTM Version 2 有些微的下降（圖十(a)(b)）。由於資料順序對模型表現並沒有顯著的提升，為此在後續的模型實驗就不再繼續打亂資料順序。

(a) Accuracy

(b) Loss

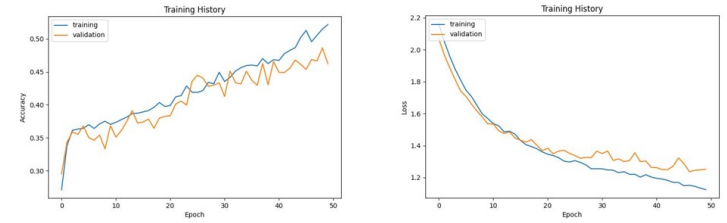


(c) 模型架構圖

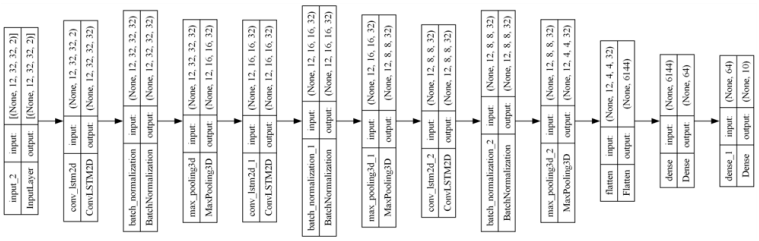
圖十：ConvLSTM Version 3 Learning Curve(a)(b)與模型架構(c)

6. ConvLSTM Version 4

在更改模型架構以及調整資料順序但沒有顯著提升表現後，最後推測可能是 optimizer 的問題。為此，根據經驗，將原先的 Adam 更換為 Adamax，參數量為 782,026（圖十一(c)）。最後在 validation set 上得到 accuracy 為 0.4623，除了 learning curve 的表現相對平整很多，效能對比前幾個版本也有顯著提升，不過仍舊是沒很令人滿意（圖十一(a)(b)）。ConvLSTM 到現階段的效能仍然無法達到要求，推測是因為 LSTM 的 Back-propagation Through Time 導致模型容易發生 gradient vanishing 或者 gradient explosion，使得訓練過程十分不穩定，且由於 CNN 的緣故導致結果容易發生 overfitting。在小組成員的討論後，決定不再測試 ConvLSTM，轉而試驗其他的模型。



(a) Accuracy (b) Loss

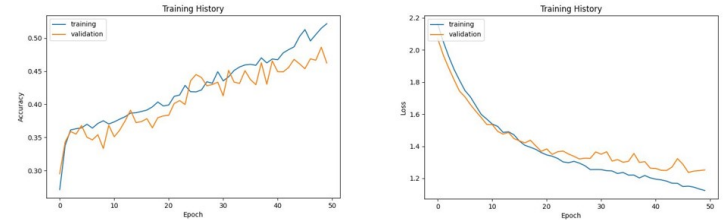


(c) 模型架構圖

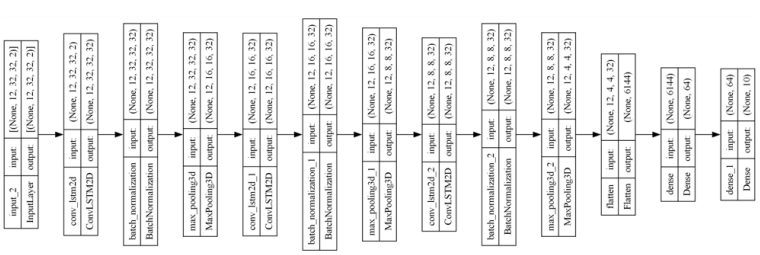
圖十一：ConvLSTM Version 4 Learning Curve(a)(b)與模型架構(c)

7. R(2+1)D CNN Version 1

再尋找各式各樣不同的模型後，最後決定使用 R(2+1)D CNN 這個模型[2]，其架構主要使用 Conv3D 的運算方法，且參考了 Depthwise Separable Convolution[3]、ResNet 的概念[4]：將 Conv3D 拆成兩個步驟，先對 Spatio 部分進行運算，再針對 Temporal 部分進行運算，並加入了 skip connection。基於上述的想法，最後建出了 R(2+1)D CNN Version 1 這個模型，總參數量為 450,154（圖十二(c)）。最後在 validation set 上得到 accuracy 為 0.6861，從 learning curve 可以發現，雖然模型訓練到最後面有明顯 overfitting 的情況，但是模型表現十分的不錯，且參數量也不多，因此這個模型有其發展性（圖十二(a)(b)）。



(a) Accuracy (b) Loss

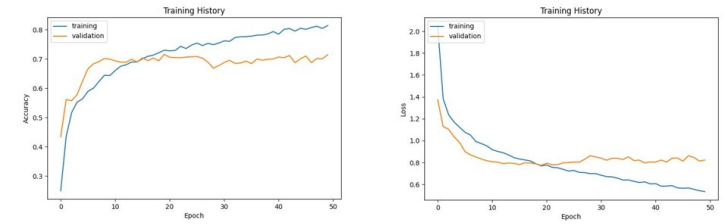


(c) 模型架構圖

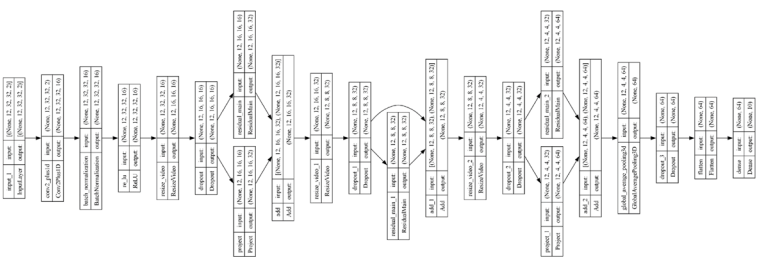
圖十二：R(2+1)D CNN Version 1 Learning curve(a)(b)與模型架構(c)

8. R(2+1)D CNN Version 2

由於上一版的模型雖然效能不錯，但是由於有 overfitting 的現象，為此本次模型加入了 dropout layer，dropout rate 為 0.3，總參數量為 431,434（圖十三(c)）。最後在 validation set 上得到 accuracy 為 0.7143，從 learning curve 也可以發現對比 R(2+1)D CNN Version 1，overfittinig 的情況確實有改善（圖十三(a)(b)）。



(a) Accuracy (b)Loss

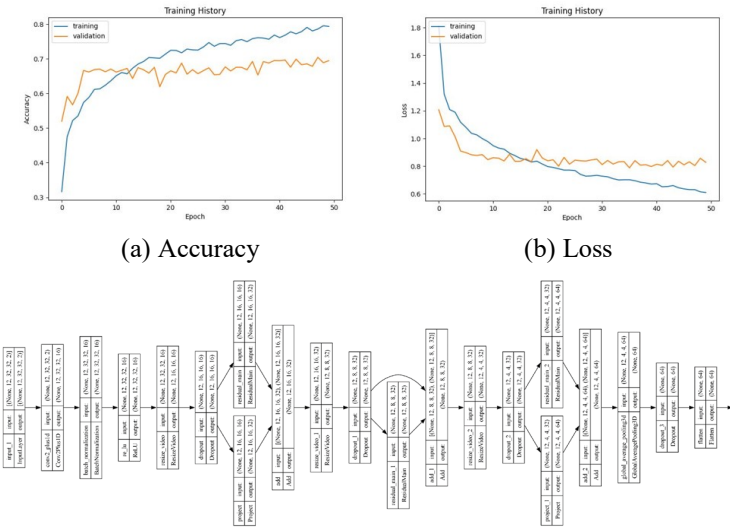


(c) 模型架構圖

圖十三：R(2+1)D CNN Version 2 Learning Curve(a)(b)與模型架構(c)

9. R(2+1)D CNN Version 3

在 R(2+1)D CNN Version 2 表現其實已經相當的不錯了，且參數量相較前面幾個模型，也減少了不少，不過考量到模型實際在跟毫米波套件串接時需要一直不斷的讀取資料並預測，在這樣的參數量下，電腦可能無法負荷。為此，在經過小組成員討論過後，決定測試看看在降低參數量的同時，能否保有不錯的效能表現。最終的總參數量降到了 126,250（圖十四(c)），並在 validation set 上得到 accuracy 為 0.7265，而 learning curve 上，雖然還是有觀察到 overfitting 的現象，但是相較於 R(2+1)D CNN Version 2 而言，並沒有明顯加劇的情況，而且效能表現也差不多，甚至更好（圖十四(a)(b)）。不過礙於時間因素所以R(2+1)D的CNN模型就只測試到這邊。



(a) Accuracy (b) Loss

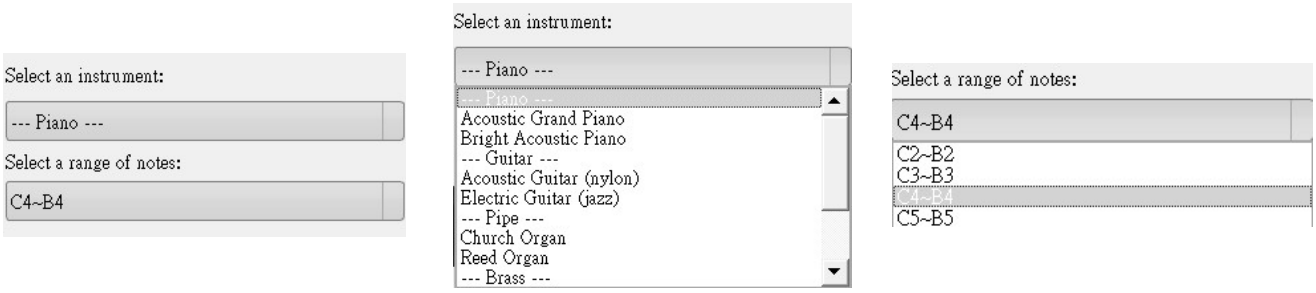
(c) 模型架構圖

圖十四：R(2+1)D CNN Version 3 Learning Curve(a)(b)與模型架構(c)

綜觀上述的9個模型後，在效能表現，CNN Version 1跟R(2+1)D CNN Version 3這兩個模型都分十分不錯，甚至毫無差別，不過從參數量的角度來看， CNN Version 1 的參數量遠大於 R(2+1)D CNN Version 3，因此最後選擇使用 R(2+1)D CNN Version 3 來當最終的模型並用於與系統串接，並將 train set 以及 validation set 作為 train set，並使用 test set 進行測試，得到 0.61 的 accuracy。

F. GUI 介面

原先在系統的設計上，在模型預測出的結果後，只使用鋼琴的聲音，後來發現 PyGame 可以透過調用 Windows 的合成器使用多種不同的聲音，且可以改變聲音的音域，因此最後在功能上新增了可以選用不同聲音。此外，為了使用者方便，還增加了 GUI 介面（圖十五(a)），使使用者可以更直觀的變更樂器聲音以及音域（圖十五(b)(c)）。



(a) 主要的 GUI (b) 樂器的選擇 (c) 音域的選擇

圖十五: 主要介面(a)與清單選擇(b)(c)

G. Difficultly

整個過程其實面臨了許多的困難，最主要的困難是電腦的記憶體以及算力有限，當一筆資料的總 frame 太高，電腦處理會非常花時間，為了處理這樣的情況，最後影片的總 frame 數才設為 190。而因為一個影片同時包含多種 label 比較符合實際使用情況，因此在收集資料上就需要在 190 個 frame 裡面做出多種動作，而這也導致收集資料的困難度非常高，容易發生操作失誤（平均一個動作／換動作只有不到一秒的反應時間）。此外，也有可能因為在收集資料過程中發生太多失誤，進而導致模型在學習過程中學到了錯誤的資訊，進而導致效能不佳。遠端工作也是挑戰之一，由於官方的 sample code 經過壓縮並傳至其他台電腦，其中的 dll.檔案會被系統阻擋而使得程式碼沒法執行。最後，因為音樂的演奏是一項精細的動作，使用毫米波雷達來偵測可能會比較難收集到手部位置精細的變化，造成資料集品質沒有想像中好，或是在實際 demo 時沒法有效的捕捉手部資訊，導致模型無法正確預測。

V. CONCLUSION

這次的實驗中，合計測試了三種不同的主要架構，分別有 3D CNN、ConvLSTM 以及 R(2+1) D CNN，在這三種不同的架構中，可以觀察到 CNN 架構本身容易出現 overfitting 的現象，如果要透過增加參數量來提高表現，則也需要加入 dropout layer 或是增加資料量；ConvLSTM 的訓練曲線與效能特別奇怪，十分容易發生 gradient vanishing 或者 gradient explosion，且有明顯訓練不起來的狀況，可能是因為 LSTM 的 Back-propagation Through Time、CNN 容易 overfitting 的問題，抑或是在模型參數與訓練方式還有需要改進的地方。R(2+1) D CNN 的架構，其表現與 3D CNN 差不多，最好的表現都有 0.7 左右，但 3D CNN 的模型在 learning curve 相較於 R(2+1) D CNN，曲線較不穩定，且參數量也比 R(2+1) D CNN 多，推測可能 3D CNN 模型無法很有效抓取到微小的動作變化，而 Depthwise Separable Convolutions 能用較少的參數量捕捉到較細微的變化，所以最後選用的模型是來自於 R(2+1) D CNN 的架構，並在 test set 上得到 0.61 的 accuracy。

這次在辨識手勢的模型實際部屬後的效果十分差強人意，推測是在資料收集出現一些問題或是模型上的問題，因此，資料方式上可能要再做一些改變，應採分開 label 的方式進行收集，使動作可以更加完整以及精細，也不會讓電腦在處裡上需要花太多時間。雖然這次實作出來的效果沒有很理想，但撇除模型效能不佳，其餘部分都可以正常的運作，因此，如果後續持續優化模型，應該可以有不錯的成效，而這次的實驗也對毫米波的未來提出了一個可能發展的趨勢，以及對於未來有想要做類似產品的人提出了一些可能性。

REFERENCES

[1] Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W., & Woo, W. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. Neural Information Processing Systems.

[2] Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., & Paluri, M. (2017). A Closer Look at Spatiotemporal Convolutions for Action Recognition. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 6450-6459.

[3] Chollet, F. (2016). Xception: Deep Learning with Depthwise Separable Convolutions. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1800-1807.

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[5] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lucic, M., & Schmid, C. (2021). ViViT: A Video Vision Transformer. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 6816-6826.

[6] Geman, S., Bienenstock, E. & Doursat, R. Neural networks and the bias/variance dilemma. Neural computation 4, 1-58 (1992).

[7] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J.I., Fadhel, M.A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8.