

數位語音處理概論 期末報告

Introduction to Digital Speech Processing – Final Project

Reading Report -

Decentralizing Feature Extraction with Quantum
Convolutional Neural Network for Automatic Speech
Recognition

(ICASSP, June 2021)

教授：李琳山 教授

學號：R09944072

姓名：鍾宜樺

前言

這是第一篇提出合併傳統機器學習與量子機器學習應用在語音辨識上的一篇論文，傳統機器學習模型皆採用目前的 State of the art 的深度學習模型。然而，不同於先前的深度學習論文，此篇論文加入了量子卷積的概念，將量子機器學習帶入語音辨識模型中，並獲得很好的語音辨識結果。

量子計算是我從大學一直做到研究所的研究領域，雖然研究時長看似不常，但卻是我很喜歡的研究領域。我希望更多人可以接觸到量子計算、感受到量子計算的魅力，因此我挑選了這篇論文作為我期末報告的主題。

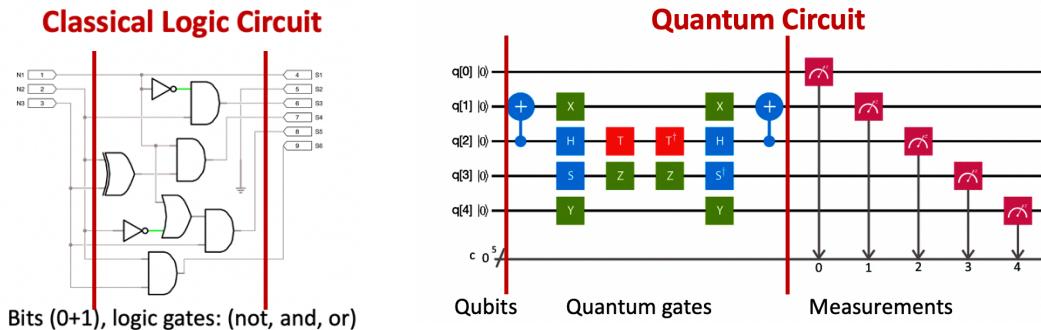
一、 簡介

本篇論文 [ref. 1] 是第一個提出同時利用：(1) 傳統電腦、(2) 量子電腦同時用於解決語音辨識的第一篇論文。將量子卷積應用於語音辨識模型的論文中，會受益於量子位元本身的存在型態，使模型參數得到保護、並透過實驗結果可發現「量子卷積確實保留可解釋的聲學特徵學習」。與當前最佳的模型 [ref. 2] 相比，作者以 [ref. 2] 作基礎的模型架構，加入量子卷積及 U-Net 專注模型所提出的 Convolutional Neural Network - Automatic Speech Recognition (ASR-QCNN) 模型顯示了具有競爭力的識別結果。

接著的報告流程如下：第二章會對量子計算做簡單的背景介紹、由於 ASR-QCNN 是根據 [ref. 2] 進行改進，因此第三章會介紹 [ref. 2] 提出的模型的詳細介紹、並於第四章針對 ASR-QCNN 模型相對 [ref. 2] 模型改進地方做詳細的介紹，同樣的部分則不再贅述。第五章會針對 ASR-QCNN 特別使用的：(1) 量子卷積的實作、(2) U-Net 專注模型進行講述。第六章則顯示 ASR-QCNN 模型的於設計實驗中的評測結果。第七章會顯示作者提供給使用者在 Google Colab 上運行此語音辨識模型特徵擷取流程與結果。第八章則對本篇論文做簡單的結論。最後於第九章提出個人對這篇論文的想法與心得。並在報告的最後面附上參考資料全名與其超連結。

二、量子背景簡介

論文在模型架構中加入了量子卷積，因此這章節簡單扼要地介紹一些量子計算的背景。



量子計算會運行在量子電路 (Quantum circuit) 上，如上右圖，量子電路與傳統電路構成相似，皆可分成三部分：(1) 輸入 (2) 邏輯閘 (3) 測量輸出。傳統電路上輸入為：位元 (bit)，量子電路上的輸入為：量子位元 (Quantum bit，簡稱：Qubit)；傳統電路上位元會經過不同的邏輯閘 (Logic gate: AND, OR NOT gates) 改變位元的高低電位，量子電路上量子位元經過量子邏輯閘 (Quantum gates) 改變量子位元存在的相位；傳統電路的最後，會測量位元為高或低電路，判斷一個位元輸出為 0 或者 1；量子電路的最後，一樣會測量每個量子位元的相位，判斷最後一個量子位元的輸出為 $|0\rangle$ 或者為 $|1\rangle$ 機率較大，進而影響該量子位元的輸出結果為 $|0\rangle$ 或 $|1\rangle$ 。

1. 量子電路三階段（初始化量子位元 → 量子位元經過量子邏輯閘改變相位 → 量子電路最後測量量子位元值）

A. 量子位元 (Quantum bit，簡稱：Qubit)

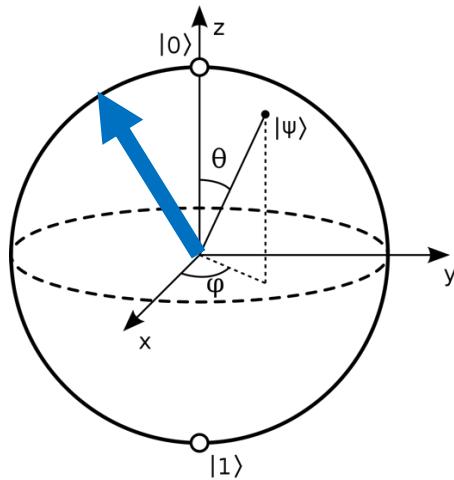
a. 量子位元不同於傳統位元在一個時間點下，只能是高電位（測量值為 1）或低電為（測量值為 0）。量子位元本身具有：(1) 量子疊加 (Superposition)、(2) 量子糾纏 (Quantum Entanglement) 等物理性質。

b. 量子疊加 (Superposition)：使得量子位元可以同時為 0 也為 1，且是以機率分佈的形式存在。會使用一個向量表是一個量子位元當下的狀態（相位）：

$$q[0] = a|0\rangle + b|1\rangle = a \begin{bmatrix} 1 \\ 0 \end{bmatrix} + b \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}, \quad a, b \in \mathbb{C}, \quad a^2 + b^2 = 1$$

其中 a^2 為該量子位元 $q[0]$ 被測量時測量到其值為 0 的機率、 b^2

則為該量子位元被測量時為 1 的機率。因此 $q[0] = a|0\rangle + b|1\rangle$ 即表示一個量子位元當時存在的形式，類正比於有 a 的機率測量時為 0， b 的機率測量時為 1。此外 a 與 b 皆為複數 (complex number) 換言之量子位元會存在於一個複數空間。因此除了使用公式表是一個量子位元當下的存在的相位外，也可以使用布洛赫球面 (Bloch Sphere)，以圖形化的方式描述量子位元當下的相位：



(圖片取自 wiki: Bloch Sphere)

會用三個角度分別描述該量子位元當下在不同方向上的相位。

當一個系統中有許多量子位元時，我們可以刻意做到（像是讓兩個量子位元之間相連一個兩單位的量子邏輯閘）讓量子位元之間產生糾纏，又稱為量子糾纏。

量子位元之間產生糾纏，以電腦科學的角度去理解就是：一個量子位元代表兩個傳統位元的情況（因為一個量子位元可以同時為 0 又為 1）。因此我們有 N 個量子位元時，即是同時測量 2^N 種組合的情況。如此一來像是 NP-hard、NP Complete 等，在傳統電腦上難以解決（需要大量計算資源、計算時間）的問題，在量子電腦上使用量子位元作為計算單元時，就可以只需要跑一次量子演算法，就同時測量 2^N 種組合 (state) 的結果。並且量子演算法的結果會以機率分佈的形式存在，其中通常以結果中出現機率最高的量子相位為所求的最佳解。舉例來說，有兩個量子位元 $q[0] = a|0\rangle + b|1\rangle$ 、 $q[1] = c|0\rangle + d|1\rangle$ ，當這兩個量子位元產生量子糾纏時，兩個量子位元存在的機率會相互被對方影響，將此系統公式化表示為 ($2^2=4$ 種狀況，使用線性代數中的張量積 \otimes 表示法)：

$$\begin{aligned}
q[0] \otimes q[1] &= (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) \\
&= (ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle) \\
&= ac \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + ad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + bc \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + bd \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}
\end{aligned}$$

c. 此外，量子位元還具有一個性質：量子位元在測量的時候量子會崩毀。簡單的理解這件事：量子位元如同薛丁格的貓一樣，在我們還沒有測量量子位元時，量子位元可以是 0 也可以是 1 (薛丁格的貓可以是存活的也可以是死亡的)，然而一旦我們對一個量子位元進行量測其值 (打開箱子查看薛丁格的貓究竟是存活還是死亡狀態) 時，量子系統就會崩毀 (collapse)，量子系統會從原本機率分佈的形式坍塌並限縮成一種狀況，通常會限縮成尚未坍塌前出現機率最大的形式 (我們確定每天都有餵食貓咪，那麼貓咪就有比較大的機率是以存活的形式存在，一旦我們打開箱子看貓咪究竟是存活還是死亡時，我們只會看到貓咪以一種狀態存在，即存活或死亡，但我們可以依照我們先前照顧貓咪的方式預測貓咪有比較大的機率是存活的，然而貓咪可能也有突然疾病過世的機率，只是發生機率較小，但不能否認貓咪仍有機率是死亡的)。

B. 量子邏輯閘，下圖為常見的量子邏輯閘，量子位元經過量子邏輯閘後，改變量子位元當前的相位：

Operator	Gate(s)	Matrix
Pauli-X (X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

(取自 wiki: Quantum gates)

量子邏輯閘皆可寫成線性代數中的矩陣形式，如上表所示。用例子認識量子邏輯閘，假設我們擁有一個量子位元： $q[0] = a|0\rangle + b|1\rangle$ ，當此量

量子位元經過一個量子邏輯閘 X-gate，X-gate 的矩陣形式如上表所示為：

$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ ，因此當 X 邏輯閘應用在量子位元 $q[0]$ ，其實就是矩陣乘法：

$$X \cdot q[0] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} b \\ a \end{bmatrix}$$

這時候我們可以發現，當 $q[0]$ 經過 X 邏輯閘後， $q[0]$ 被測量時為 0 和為 1 的機率會被顛倒，若原本該量子位元被測量值為 0 的機率比較大，經過 X 邏輯閘後會變成為 1 的機率比較大。故可以知道 X 邏輯閘其實就代表著傳統邏輯電路上 NOT-gate 的意義。

另外，我們也可以帶一次矩陣乘法發現量子邏輯閘中的 CNOT-gate，其時就帶表傳統電路上的 XOR-gate。當控制位元為 1 的時候，目標位元的為 0 和為 1 的機率也會相反；然而，當控制位元 0 的時，目標位元的機率分佈則不改變，公式如下：

$$\begin{aligned} q[0] \otimes q[1] &= (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) \\ &= ac \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + ad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + bc \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + bd \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix} \end{aligned}$$

應用 CNOT 邏輯閘：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ \textcolor{blue}{bc} \\ \textcolor{blue}{bd} \end{bmatrix}$$

2. 整理上述，換言之運行一個量子電路就是在做線性代數中的矩陣乘法。在線性代數中，我們又可以知道：矩陣乘法又可以被理解：線性變換 (linear function)、即將原本向量做線性組合，得到新的向量的線性組合。而該線性變換中的 linear function，可以被應用在深度學習中使用 CNN 做特徵抓取時 kernel 的角色，換言之，跑一個量子電路，其實就可以被應用在機器學習中做編碼 (encode) 的部分，這部分的研究領域統稱為 Quantum Machine Learning，進而延伸出像是量子卷機 (Quanvolutional, QCNN) 等相關名詞，本篇論文便是應用量子卷積在語音辨識的深度學習模型。

3. 量子計算相對於傳統計算，在電腦科學領域的優勢：

- A. 使用量子演算法對 raw data 做編碼，可以保證遵守 Data Protection Regulation, GDPR 的規範。根據上述已知，由於量子一旦被測量則該量子系統就會坍塌 (collapse)，從 2^N 種狀況的機率分佈收斂成 1 種狀態。因此，若有不法人士想要偷竊用量子狀態中的資料，就會造成系統崩塌即刻會被發現。

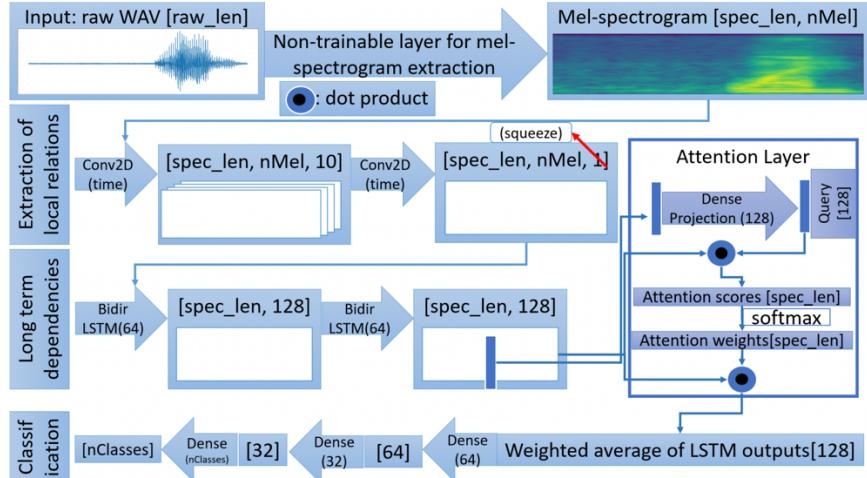
- B. 量子位元本身具有量子疊加與量子糾纏的物理性質， N 個量子位元即可表示 2^N 種情況，因此每跑一次量子演算法就等同於處理 2^N 種可能性，故可以大量減少超參數的數量。
- C. 當前量子演算法中，有特定的量子演算法（由特定的量子電路構成）是專門做 search based 的演算法，像是著名的 Grover Algorithm，可以放大指定的 state (共有 2^N 種 states)，在 Quanvolutional 中則可以選擇放大輸入中特定的特徵。

三、相關研究、論文

提及 QCNN-ASR 模型架構前，先介紹 QCNN-ASR 的前身“Attention RNN”。由里約熱內盧聯邦大學 (Universidade Federal do Rio de Janeiro) 於 2018 年發表在 arXiv : A neural attention model for speech command recognition [ref. 2] 中提出的模型。

此開源的 RNN 模型專用於語音辨識 (spoken word recognition)，且結果優於其他基於 DNN 建立的模型，包括：DS-CNN 和 ResNet。此外，該論文將 attention weight 做成可以視覺化的結果，並提出將注意力力模型 (Attention-based model) 可視化後，使得語音識別模型具可解釋性後、並提高其正確率。

該模型架構如上，且有以下特徵：



Step 1. 該模型的輸入為限制時間長度為一秒的語音 raw WAV

Step 2. 將原本不可訓練的語音，轉成標示出特徵的梅爾頻譜圖。論文中提及，梅爾頻譜圖轉換是用 Kapre；然而，我看了論文開源的程式 [ref. 2-I] 應該是使用使用 Librosa library，對輸入 raw WAV 提取特徵，故將兩者都列在此。

NOTE 1. 梅爾頻譜圖可作語音辨識系統中的特徵質觀察，例如：自動辨認一個人透過電話說的數字、抑或是作為用來判斷語音訊號的發話者是誰的聲紋辨識 (Speaker Recognition) 等 (From wiki. 梅爾頻譜圖)。

NOTE 2. Kapre 是 Keras 中的音頻預處理器。

NOTE 3. Librosa 是 Python 的 package：專門做分析聲音訊號的模組。有對輸入音頻做時頻處理、特徵提取、繪製聲音的梅爾頻譜圖、…，等功能。

Step 3. 得到梅爾頻譜圖後要從頻譜圖中抓取特徵關聯性 (local relations)。頻譜圖會經過兩層二維卷積層 (2D convolution layer)。

I. 卷積層的用意是來提取頻譜圖中的特徵：

- A. 其中卷積的核 (kernel) 大小廣義來說：會以觀察的特徵值得細微程度做決定，當提取的特徵值較大時，則核大小取大；當觀察的特徵值較小時，則核的大小會取較小便利於取得較細微的特徵。
- B. 本篇論文的卷積的核 (kernel) 大小：前者為 10 後者為 1。為的是在第一層卷積時，是提取頻譜中大範圍的特徵，第二層再提取其中較細緻的特徵。另外，第一層卷積核大小取較大的原因是為了防止頻譜洩漏 ([ref. 9] 討論 CNN kernel 過小造成的頻譜洩漏)。

II. 兩層 CNN 的激活函數 (activation function) 皆為：relu。

Step 4. 兩層 CNN 提取完頻譜圖的特徵後，會再經過兩層雙向的長短期記憶模型 (Long-Short Time Memory, LSTM)，

I. 於此使用長短期記憶模型的意義：由於音頻中前後具有其關連性，而 LSTM 的優勢便是保留資料長期的記憶，故在此用 LSTM 獲取音頻中雙向（向前和向後）長期音頻的關聯性 (dependencies)。

Step 5. 自 LSTM 出來後，會過 attention layer。利用任意取一輸出向量，利用投影的方式查詢輸出的向量中，哪一部分對於用來識別音頻為何指令時，擁有最高的相關性。

- I. 這部分論文中提取的是 LSTM 輸出中的「中間向量」。原因是因為語音命令 (speech command) 中預計將出現在音頻中位在中間的位置。
- II. 作者認為 LSTM 輸出的任意向量都可以使用來投影、找尋音頻相關性的評斷的原因：認為雙層 LSTM 具有能夠承載足夠長久的「記憶」的能力。

Step 6. 最後會將 LSTM 輸出的所有向量加權平均後，送入三個全連接層 DesneNet 進行分類。

- I. 三個全連接層分別為：Dense(128)，Dense(64)，以及 Dense(32)，具有 128、64、32 個隱藏神經元的全連結層，進行最後的三層分類。
- II. 論文中採用 DenseNet 而非 ResNet 的原因是因為： DenseNet 的是通過特徵在通道 (CNN 的 channel，又理解成維度) 上是全連結的，因此相對 ResNet 一般是 2~3 個節點連結到下一層的一個節點而言，DenseNet 的每一層都會接受其前面所有層作為其額外的輸入。換言之，DenseNet 相對 ResNet 的優勢便是 DenseNet 可以藉由全連結做到特徵重用的 (Feature reuse) 以獲得更完整的資訊、使此論文中的模型分類更準確。

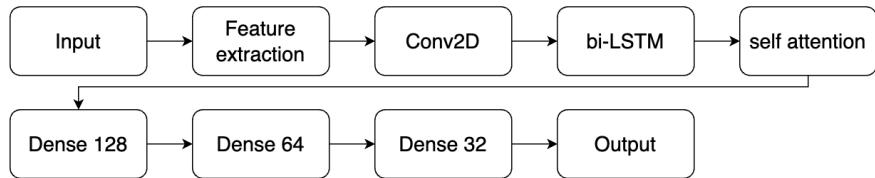
介紹完該模型後，我們簡單簡述一下此訓練此模型的重點，在讀入一個一秒的語音 raw WAV 後，此語音進入模型後，希望輸出判斷此 speech command 是哪一個指令的準確率越高越好。即讀入音檔後，經模型判斷將該語音指令分類至正確的指令集。

四、 模型架構

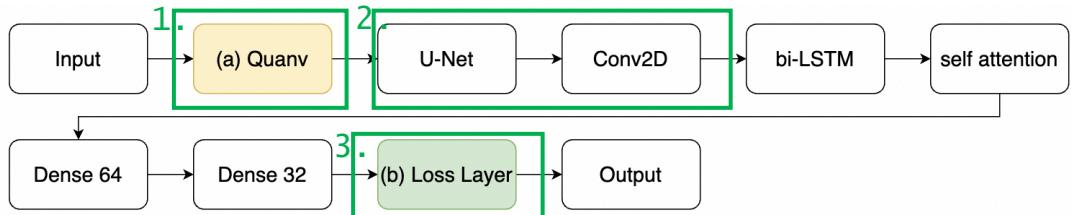
接著介紹主題論文提出的模型架構 — Convolutional Neural Network - Automatic Speech Recognition (QCNN-ASR)。

由於 QCNN-ASR 的基礎架構是以前一章節介紹的 Attention RNN 為基礎去做部分改變，因此我們直接放上兩種模型架構圖做比較。

前一章介紹的 Attention RNN 的簡化模型架構圖，如下：



接著放上 QCNN-ASR 的模型架構圖：



根據上下模型架構圖比較可發現，作者提出來的 QCNN-ASR 與 Attention RNN 不同的地方如下（上圖中綠框的部分）：

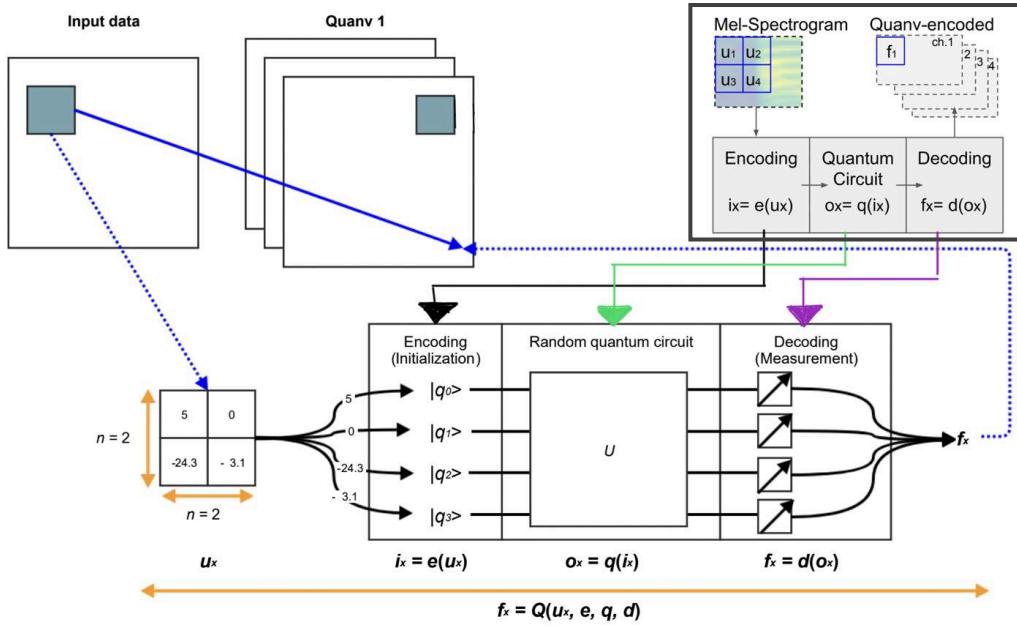
1. 特徵抓取：不同於 Attention RNN 採用開源包 (Kapre 或 Librosa) 產生音檔的梅爾頻譜圖後，就進入 Conv2D 做特徵抓取。QCNN-ASR 採用的方法
 - A. 使用開源包 (Librosa) 建造音訊檔的梅爾頻譜圖。
 - B. **(綠框 1)** 相比原架構多一層 Quantum Convolutional Layer 又稱為 (a) Quanv，作用為將梅爾頻譜圖經過量子編碼 (QuantumEncoded) 傳入量子電路，對原梅爾頻譜圖進行特徵加強與編碼。
 - C. **(綠框 2)** 在量子進入 Conv2D 前為了減少量子 (Quantum) 與經典 (Classical) 兩種不同架構中的變數轉換 (reduce variants between architecture-wise)，作者採用和前一篇論文不同的做法：使用 U-Net 一種 self-attention encoder。
2. **(綠框 3)** 自 Dense32 分類出來後，會計算損失函數 (Loss layer)，論文中使用的是：交叉熵損失函數 (cross-entropy loss)，用來驗證新增的 (a) Quanv 做特徵強化的正確率。
 - A. 損失函數這部分作者也有提到，在未來此模型做成 Large-scale 時，可以使用 Connectionist temporal classification (CTC) loss 替代交叉熵損失函數交叉熵損失函數的方式。

五、 實作

本章節將著重在 QCNN-ASR 提出的模型架構實作，由於 QCNN-ASR 提出的架構基本上與 Attention RNN 基本相同，此章節將著重講解上章節中未提及的：(1) **(綠框 1)** Quantum Convolutional Layer (本篇論文中稱為：(a) Quanv，Quanvolutional 的簡稱) 的實作、(2) **(綠框 2)** U-Net based attention 模型簡介。

整個實作都是使用 tensorflow 2.0 with CUDA 10.0. 完成，搭配 PennyLane [ref. 7] 量子電腦開源包 (PennyLane 是多倫多大學專門做 Quantum Machine Learning 的團隊所示出的開源包，可以搭配 PyTorch、TensorFlow、Strawberry Fields、Forest (Rigetti) 和 Qiskit (IBM) 等平台使用，亦可以用 CPU、GPU 或 TPU 作加速)，本份論文的開源碼如 [ref 2. II] 所示。

- ✓ Quantum Convolutional Layer (a) Quanv : 特徵加強與安全編碼的實作
其流程圖如下圖中右上圖所示：



上圖中大圖取自 [ref. 4]

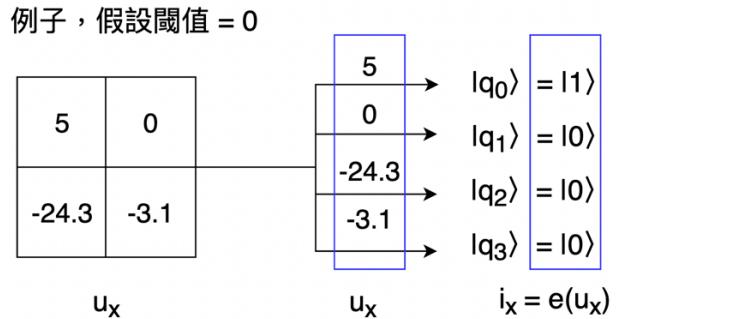
Step 1. 切割梅爾頻譜圖成量子電路的輸入單位： u_x ，一個大小為 $n \times n$ 的二維矩陣，矩陣的每格存放該梅爾頻譜圖的像素值 (pixel value)，其中 $n > 1$ 。

- I. 上圖左下角此處圖例可知 u_x 於此例子中大小為 2×2 ，即將整個梅爾頻譜圖切成 $(長度/2) \times (寬度/2)$ 個 u_x 。
- II. 在作者能使用的 NISQ 機器的量子計算位元 (qubit) 有限的情況下 (最多 16 qubits)。 u_x 在本篇論文中最大的大小只能切成 3×3 (9 qubits)。

Step 2. 初始化編碼 (Encoding Initialization)：將 u_x 的信息編碼 (encode) 到每個單獨的量子位 (qubit) 的初始狀態。初始狀態定義為 i_x ， $i_x = e(u_x)$ 。

- I. 能將 u_x 編碼成 qubit 形式的方式有很多，例如：[ref. 10]。
- II. 當今有很多方法可以將 u_x 中的一個 pixel 帶有的信息編碼成 qubit 的初始化狀態。不同的論文會採用不同的編碼方式，在本篇論文中採用很簡單的方式：
 - i. 作者設定了一個閾值 (threshold)，並對每個像素應用這個閾值，當原 pixel 值大於閾值，則將代表此 pixel 的 qubit 值初始化為 $|1\rangle$ ；反之，當像素值小於等於設定閾值，則初始化為 $|0\rangle$ 。

ii. 例子：



Step 3. 運行量子電路調整超參數(Random quantum circuit)：量子位元初始化後，量子位元運行在量子電路上面，計算量子位元經過量子邏輯閘後的結果，最後輸出量子狀態 o_x ，關係式為 $o_x = q(i_x) = q(e(u_x))$

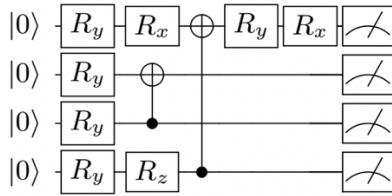
I. 這裡要敘述一下為什麼可以用運行量子電路表現和跑機器學習相同的結果：

- i. 在第二章、量子背景介紹中提到的，量子位元本身的物理特性「量子疊加」(Superposition) 能夠使得一個量子位元以機率分佈的形式存在，換言之一個量子位元同時是 0 也同時是 1，只是測量時為 0 和為 1 的機率不同，每個量子位元同時代表兩個狀況。
- ii. 此外，量子位元之間會因「量子糾纏」(Quantum Entanglement) 進而產生連結，使得所有量子位元之間具有相互關係（即一個量子位元的相位被改變時，量子位元之間的相位都會相互影響）。
- iii. 當我們有 N 個量子位元的時候，則 N 個量子位元則代表 2^N 種狀況。故，當我們運行量子演算法時，不同於傳統演算法要測量 2^N 種狀況時，演算法需要運行 2^N 次；量子演算法運行一次，即同時考量 2^N 種狀況，而其輸出則為所有狀況 (2^N 種狀況) 做為結果的機率分佈。

II. 那量子 CNN 中的量子電路代表什麼？當量子位元運行在量子電路上即是做量子位元的相位轉換，反映在傳統 CNN 上就是做超參數的調整。傳統上利用調整 CNN 上的超參數，進而改變每個 pixel 值得權重…等。在量子 CNN 上則是利用量子位元經過量子電路上的量子邏輯閘後，會改變量子的相位的方式，調整每個 pixel 的權重。

- i. 此量子電路為論文中最中選用的量自電路，其中 Rx, Ry, Rz 皆是對量子位元在布洛赫球面 (Bloch Sphere) 上三個不同的軸上對量子位元的相位做不同方向上的旋轉，而十字搭配一黑點為 CNOT

量子邏輯閘，用途與傳統電路上的 XOR 相同，當控制位元 (control bit) 為 1 的時候，目標位元 (target bit) 從 1 變為 0、或者 0 變為 1；在量子位元上的意義則是該目標位元被測量時為 $|0\rangle$ 和為 $|1\rangle$ 的機率進行調換。



(b) Deployed Quantum Circuit.

III. 於此使用 Quanv 進行特徵加強過濾掉不重要的訊息時，帶來的好處為：可以減少傳統 CNN 中的超參數的使用，如量子電路所示，每新增一個旋轉邏輯閘 (R_x, R_y, R_z) 只會多一個超參數，因此上面的電路總共只有 8 個超參數，相比於在傳統做法上可以大量減少超參數。即我們於量子電路中調整旋轉相位的角度，就是傳統 CNN 中調整超參數的過程。

Step 4. 將量子位元解碼回傳統位元表示式，再進行傳統的 CNN (Decoding for the qubit, Measurement)。

I. 這部分論文是採用目前 PennyLane 所提供的 `expval` function [ref. 11]，呼叫這個函數會自動將量子位元形式解碼回傳統位元（此為一個研究領域，如何解碼可以獲得更好的效果），數學公式定義成： $f_x = d(o_x) = d(q(e(u_x)))$ ， d 是解碼函數。

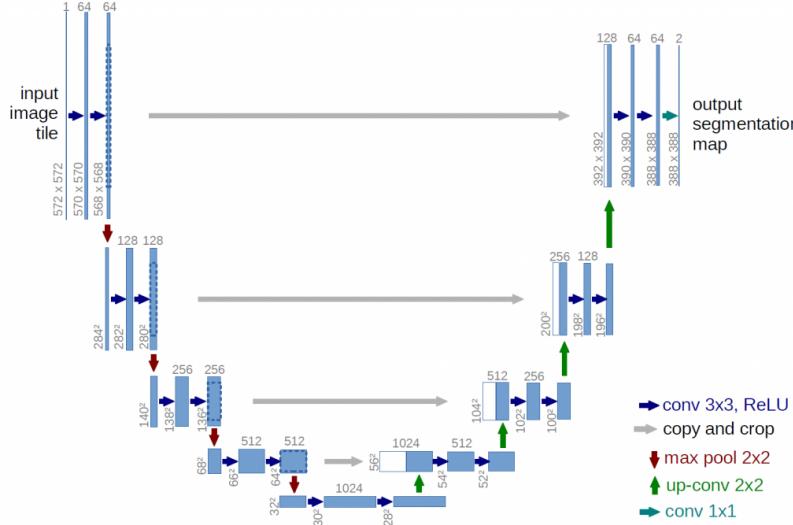
有關 (a) Quanv 這部分的實作檢驗，請看第六章、評測結果的第一小節。

✓ U-Net 注意力模型 (U-Net based attention model) 簡介

本論文提出的 ASR-QCNN 相比其前身 Attention RNN，除了上述特徵抓取的部分作法不同外，其模型第二個不同點在於走完兩層 LSTM 後，Attention RNN 會經過一層 attention layer (第三章 Step5. II) 提取重要的資訊外，ASR-QCNN 在這裡使用的是 U-Net 注意力模型。

U-Net 注意力模型：U-Net 的名字來源於當我們將其模型可視化時，其模型架構圖看起來像字母 U，如下圖所示。U-Net 的作用與一般 attention model 類似，但 U-Net 模型在模型左半部會進行降維的動作，右半部則進行升維的動作。這種降維後又升維過程的意義在於：提高輸出的分辨率，留下高分辨率特

徵、將異常的特徵點進行消除，使得信息產生更精確的輸出。(此段參考自 [ref. 13])。

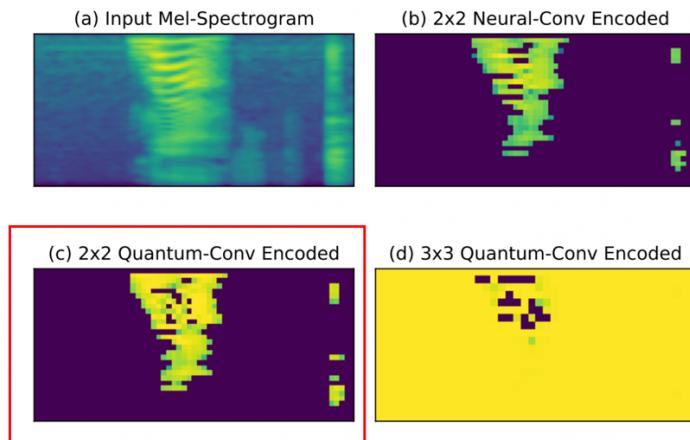


圖片取自 [ref. 13]

六、評測結果

評測實驗為測試新的模型架構的可行性，因此解決的語音辨識問題僅使用 Google Speech Command V1 與 V2 有限制的字典集。Google Speech Command V1 為平時最常見的指令集，共有十個 Classes，包含：['left', 'go', 'yes', 'down', 'up', 'on', 'right', 'no', 'off', 'stop']，共 11165 筆訓練資料 (training data)、6500 筆測驗資料 (testing data)，且資料集中有白噪音污染 (background white noise setup)。

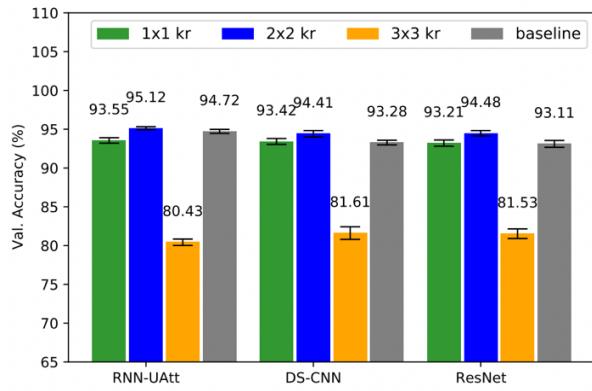
實驗一、檢驗 ASR-QCNN 模型的可行性，輸入指令為‘yes’：



1. ASR-QCNN 於上圖左下角 (c) 2*2 Quantum-Conv Encoded，我們可以發現 ASR-QCNN 可成功抓出輸入的梅爾頻譜圖上的特徵，證明使用 Quanvolutional 抓取梅爾頻譜圖的特徵值是可行的。

2. 比較 ASR-QCNN 與其前身 Attention RNN 做特徵抓取的效果如上圖中
(b) 2*2 Neural-Conv Encoded 與 (c) 2*2 Quantum-Conv Encoded。可發現兩種方法皆可以抓取出輸入的梅爾頻譜圖的大致特徵輪廓外。然而，ASR-QCNN 相對於 Attention RNN 更能抓取到梅爾頻譜圖中更確切的特徵。
3. 同樣作為 QCNN 模型，比較使用不同的核大小，分別為：(c) 2*2 核、(d) 3*3 核大小得到的結果，可發現較大的核在 QCNN 中並不一定較能找到更確切的特徵值。

A. 作者根據 QCNN 模型中核大小比較，另外做實驗使用 QCNN 模型在使用不同的核大小抓取特徵後，串連三種不同的模型，分別為：(1) RNN-UAtt、(2) DS-CNN、(3) ResNet 與訓練出來的深度學習聲學模型 (DNN acoustic models) 判定指令正確性結果如下圖：



B. 作者認為使用 3*3 大小的量子核會使得特徵抓取效果不佳的原因為：從前一個實驗可發現，3*3 大小的核在特徵抓取通常過於稀疏且沒有區分度，進而造成模型效果不好。故得使用較大的 QCNN 核大小，並不一定會保證提高模型性能。

實驗二、語音辨識識別實驗

在確認 ASR-QCNN 的可行性後，作者比較 RNN_{ATT} (此論文的前身：
Attention RNN)、Conv + RNN_{ATT} (在 RNN_{ATT} 模型前加上卷積做特徵抓取的模型)、Quanv + RNN_{ATT} (本論文提出的 ASR-QCNN 模型中的 U-Net 專注模型改成使用一般的專注模型)、RNN_{UATT} (Attention RNN 中的專注模型改成使用 U-Net 專注模型)、Conv + RNN_{UATT}、以及 Quanv + RNN_{UATT} (本論文提出的方法：ASR-QCNN) 等六種模型。在 Google Commands datasets 中再有白噪音的環境下判斷語音指令的正確率，結果如下表所示：

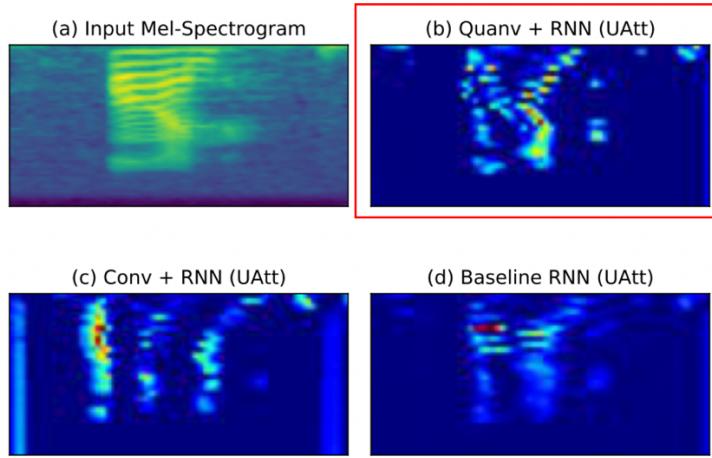
Table 2: Comparisons of spoken-term recognition on Google Commands dataset with the noise setting [29] for classification accuracy (Acc) \pm standard deviation. The additional convolution (conv) and quantum convolution (quadv) layer have the same 2×2 kernel size.

Model	Acc. (\uparrow)	Parameters (Memory) (\downarrow)
RNN _{Att} [26]	94.21 \pm 0.30	170,915 (32-bits)
Conv + RNN _{Att}	94.32 \pm 0.26	174,975 (32-bits)
Quadv + RNN _{Att}	94.75 \pm 0.17	174,955 (32-bits) + 4 (qubits)
RNN _{UAtt}	94.72 \pm 0.23	176,535 (32-bits)
Conv + RNN _{UAtt}	94.74 \pm 0.25	180,595 (32-bits)
Quadv + RNN _{UAtt}	95.12 \pm 0.18	180,575 (32-bits) + 4 (qubits)

1. 作者認為從實驗看來，具有量子卷積的識別模型顯示出比基線模型 (RNN_{ATT}) 更好的準確性。證明了此論文提出在前模型架構中新增量子卷積層(**第四章、模型架構中的綠框1**)，為有效增進模型確率的方法。
2. 帶有 U-Net 專注模型的 RNN_{UATT} 相比原本的 RNN_{ATT} 擁有更高的準確率 ($94.21\% \pm 0.30$ vs $94.72\% \pm 0.23$)。證明了此論文提出在前模型架構中，將原本的專注模型更換為 U-Net 專注模型 (**第四章、模型架構中的綠框2**)，為有效增進模型確率的方法。
3. 從上表中可觀察到，作者提出的最新的模型架構的準確率為所有模型架構中，正確率最高的模型，正確率為 $95.12\% \pm 0.18$ 。

實驗三、視覺化 ASR-QCNN 模型討論 QCNN 為何能獲得較好的結果的原因，輸入指令為‘on’：

作者在論文的最後使用 Class Activation Mapping (CAM) 這個技術，將特徵視覺化以便觀察不同的模型會專注在梅爾頻譜圖上的哪個部分，結果如下：



1. 如上圖所示，作者提出的 ASR-QCNN 於上圖 (b) 中，比僅具有捲積層和基線模型的 RNN 模型，使用量子卷基層能夠學習更多相關和更豐富的聲學

特徵。根據 CAM 顯示，激活的隱藏神經元在學習時可以識別梅爾頻譜圖中相關的低頻模式。

七、 Google Colab – 程式 Demo

作者們也有在 Google Colab 中提供大家可以直接受 Colab 使用他們建造的 ASR-QCNN 模型，Google Colab 網址如 [Ref 1.III]。我有上去簡單的測驗一下感覺得很有趣。除了使用 Colab 不需要自己裝載該環境外、也可以簡單的使用作者釋出的 Demo 程式，看一下使用 ASR-QCNN 抓取出的我聲音的特徵圖。

Step 1. 將程式運行後，程式會錄一秒鐘的聲音：

```

  from google.colab import output as colab_output
  from base64 import b64decode
  from io import BytesIO
  from pydub import AudioSegment
  import IPython.display as ipd
  import librosa

  RECORD = """
  const sleep = time => new Promise(resolve => setTimeout(resolve, time))
  const b2text = blob => new Promise(resolve => {
    const reader = new FileReader()
    reader.onloadend = e => resolve(e.srcElement.result)
    reader.readAsDataURL(blob)
  })
  var record = time => new Promise(async resolve => {
    stream = await navigator.mediaDevices.getUserMedia({ audio: true })
    recorder = new MediaRecorder(stream)
    chunks = []
    recorder.ondataavailable = e => chunks.push(e.data)
    recorder.start()
    await sleep(time)
    recorder.stop() =>{
      blob = new Blob(chunks)
      text = await b2text(blob)
      resolve(text)
    }
    recorder.stop()
  })
  """

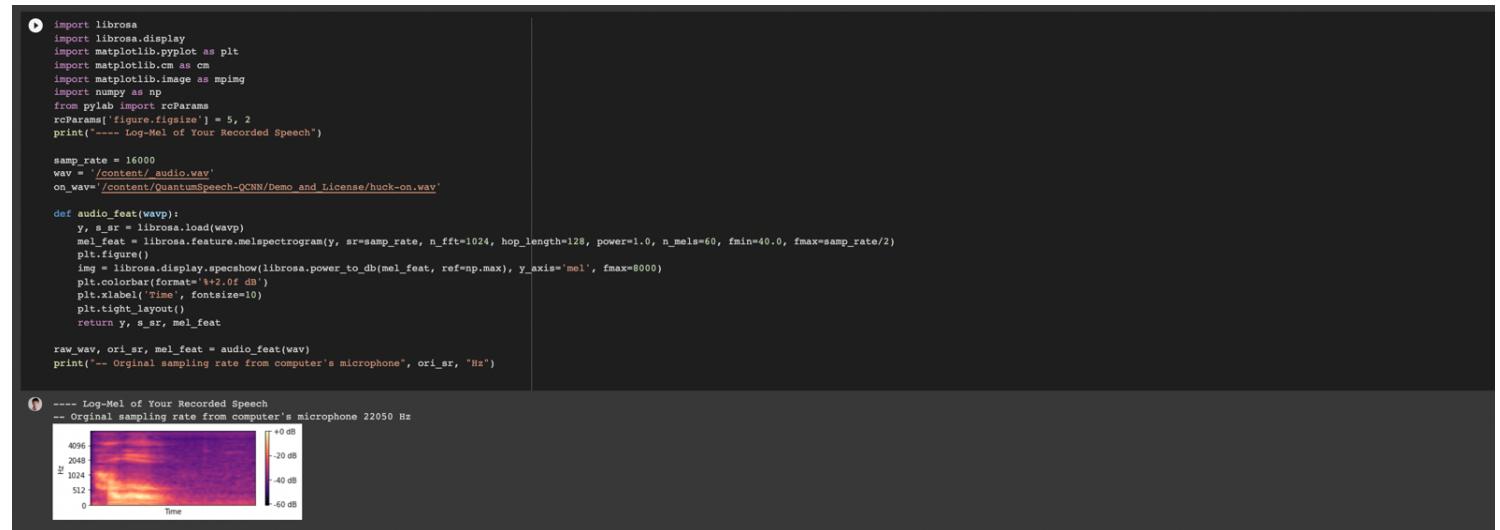
  def record(seconds=1):
    display(ipd.Javascript(RECORD))
    print("Recording started for (%seconds) seconds.")
    s = colab_output.eval_js("record(%d)" % (seconds * 2000))
    print("Recording ended.")
    b = b64decode(s.split(",")[-1])

    fileformat = "wav"
    filename = "%audio.%fileformat"
    AudioSegment.from_file(BytesIO(b)).export(filename, format=fileformat)
    return librosa.load(filename)

    waveform, sample_rate = record()
    ipd.Audio(waveform, rate=sample_rate)

  Recording started for 1 seconds.
  Recording ended.
  
```

Step 2. 接著會使用 librosa 計算出梅爾頻譜圖如下圖左下角所示：



Step 3. 計算完梅爾頻譜圖後會使用量子電路進行特徵抓取的動作：

Define Random Circuit Layer

Let's see how quantum convolution works

▶ ↳ 2 個隱藏的儲藏格

Feature Encoding by Random Quantum Circuit

```
[ ] q_mel_feat = quanv(mel_feat)
```

Step 4. 最後運行出其 ASR-QCNN 模型對使用者的聲音的特徵抓取圖：

Visualization for Each Convolutional Channel

```
❶ n_samples = 1
n_channels = 4

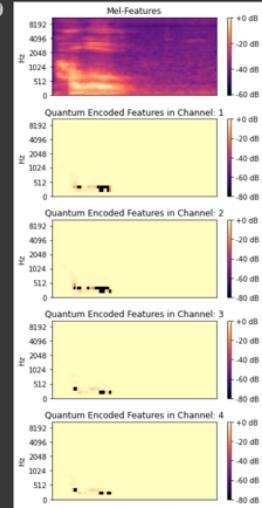
from pylab import rcParams
rcParams['figure.figsize'] = 5, 10

def channel_show(mel_feat, q_mel_feat):
    plt.figure()
    plt.subplot(5, 1, 1)
    librosa.display.specshow(librosa.power_to_db(mel_feat, ref=np.max), y_axis='mel')
    plt.colorbar(format='%.2f dB')
    plt.title('Mel-Features')

    # Plot all output channels
    for c in range(n_channels):
        plt.subplot(5, 1, c + 2)
        librosa.display.specshow(librosa.power_to_db(q_mel_feat[:, :, c], ref=np.max), y_axis='mel')
        plt.colorbar(format='%.2f dB')
        plt.title("Quantum Encoded Features in Channel: " + str(c + 1))

    plt.tight_layout()
    plt.show()

channel_show(mel_feat, q_mel_feat)
```



八、結論

這篇論文提出了新的特徵提取方法，是第一個將量子卷積應用於語音辨識模型的論文。使用量子卷積做特徵提取有以下好處：(1) 利用量子型態儲存資訊，有助於模型參數保護，(2) 我們通過也驗證了「量子卷積確實保留可解釋的聲學特徵學習」。

另外，作者提出的 ASR-QCNN 模型顯示了具有競爭力的識別結果，與當前的 state of the art Attention RNN 相比，也證明量子卷積在學習時能做到性能穩定的口語識別。

九、個人想法與心得

首先提及對論文的想法。作者在會議的演講 [Ref. 1-I] 中有提到：(1) 因為現實中量子電腦量子電腦計算元數量只有 9 量子位元的限制下，因而使模型所使用的核大小最大只能為 3×3 、(2) 此外在建立隨機量子電路的時候不能使用太多 CNOT 邏輯閘，因為 CNOT 邏輯閘在當前的量子電腦中是產生最多錯誤的量子邏輯閘。

我認為這兩個問題都可以在量子電腦的模擬器上面得到解決，雖然量子電腦的模擬器現在受限於「使用傳統電腦模擬量子電腦需要極大的記憶體」這個問題，但目前能夠做到 50 qubits 以上的開源量子模擬器比比皆是，例如：QISKit (IBM), Cirq (google), Forest (Rigetti and Oxford), ProjectQ (ETH Zurich) 等，都可以免費租用，或許將此研究運行在量子模擬器上，可以讓研究有更大的發展空間。

但不可否認，作者很厲害能夠將量子應用在語音辨識模型上面，除了確認量子的確可行之外、更獲得了比傳統 RNN 更好的準確率，著實讓人驚艷。並且作者在特別強調的：本篇論文提出的模型不只可用在語音辨識，也可以用在像是圖形辨識 [Ref. 14] 等領域，我相信這個模型也可應用在更多的領域上。

這學期也是我第一次接觸語音辨識，從期中考的成績感受到自己並沒有真正學懂這門學科的感覺，但又希望自己不是白白修一門課，希望能夠從 DSP 這門課讓自己產生一些對於學術的熱情，因此我從期中考後我就在挑選期末報告的主題。寫報告的時候覺得要整理不同領域的東西感覺很累，整理完這篇論文的前身 Attention-RNN 實在很想交期末報告了。但又真的很希望大家可以看見量子計算正在漸漸普及到各個研究領域中，甚至是今天的主題語音辨識裡。因此最後還是很努力地寫完了這份報告。很努力的想使用較短的篇幅將大家不熟悉的量子領域解釋清楚。總之最後將我的研究領域量子計算加入期末報告中真的讓我覺得很開心，也希望教授如果對報告有任何覺得我寫不清楚的地方，我也很願意增添更多的解釋與敘述！

¶ Reference

1. 論文本文 - Decentralizing Feature Extraction with Quantum Convolutional Neural Network for Automatic Speech Recognition
(Link: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9413453>)
 - I. Talk - ICASSP 21 - Decentralizing Feature Extraction with Quantum CNN for Automatic Speech Recognition
(Link: <https://www.youtube.com/watch?v=ZigIaFFFUhw>)
 - II. Github 開源程式碼
(Link: <https://github.com/huckiyang/QuantumSpeech-QCNN>)
 - III. Colab (範例) Link:
<https://colab.research.google.com/drive/1gHawQf6G1xRvb45OObe5fOyRdAPB5pkq#scrollTo=QtmZnoCmEjdG>
2. A neural attention model for speech command recognition (arXiv)
(Link: <https://arxiv.org/abs/1808.08929>)
 - I. Github 開源程式碼
(Link: <https://github.com/douglas125/SpeechCmdRecognition>)
3. Federated Learning of N-gram Language Models (arXiv)
(Link: <https://arxiv.org/abs/1910.03432>)
4. Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits (arXiv)
(Link: <https://arxiv.org/pdf/1904.04767.pdf>)
5. A Speech Recognition Based on Quantum Neural Networks Trained by IPSO (AICI 2009)
(Link: <https://ieeexplore.ieee.org/document/5375871>)
6. An introduction to quantum machine learning (arXiv)
(Link: <https://arxiv.org/abs/1409.3097>)
7. PennyLane (Link:
<https://medium.com/xanaduai/training-quantum-neural-networks-with-pennylane-pytorch-and-tensorflow-c669108118cc>)
8. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization (arXiv)
(Link: <https://arxiv.org/abs/1610.02391>)
9. Spectral Leakage and Rethinking the Kernel Size in CNNs (arXiv)
(Link: <https://arxiv.org/abs/2101.10143>)

10. Encoding classical data into a quantum computer (arXiv)
(Link: <https://arxiv.org/abs/2107.09155>)
11. qml.expval
(Link: <https://pennylane.readthedocs.io/en/stable/code/api/pennylane.expval.html>)
12. Grover Algorithm
(Link: https://en.wikipedia.org/wiki/Grover%27s_algorithm)
13. Deep Learning for Image Segmentation: U-Net Architecture
(Link: <https://heartbeat.comet.ml/deep-learning-for-image-segmentation-u-net-architecture-ff17f6e4c1cf>)
14. Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits (arXiv)
(Link: <https://arxiv.org/abs/1904.04767>)