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 Thesis

用於邊緣裝置之低耗能關鍵詞擷取系統的研究

A Study on Small-footprint Keyword Spotting for Edge Devices

林傳祐
Chuan-You Lin

指導教授：張智星博士
Advisor: Jyh-Shing Roger Jang, Ph.D.

中華民國 109 年 6 月
June, 2020

doi:10.6342/NTU202001860

國立臺灣大學碩士學位論文
口試委員會審定書

用於邊緣裝置之低耗能關鍵詞擷取系統的研究

A Study on Small-footprint Keyword Spotting for Edge Devices

本論文係林傳祐君（學號R07922104）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 109 年 6 月 30 日承下列考試委員審查通過及口試及格，特此證明

口試委員：張智星

（指導教授）

王新民　　李宏毅

林奕辰

系 主 任　莊永裕

誌謝

首先我要感謝我的指導教授張智星老師，開啟我的碩士班生活，從進入實驗室開始就引領著我，對語音領域從一開始的懵懵懂懂，經過實驗、修課、研究一步步地完成這篇論文，在遇到瓶頸的時候也會給予我建議，幫助我找出方向。此外，老師也會教導我們人生的道理，對於我們未來出社會有很大的幫助；平時也會帶學生一起運動，讓大家不會總是窩在電腦前，關心我們的身體健康。謝謝老師帶給我充實的碩士班生活。

感謝實驗室的所有同學，讓實驗室在研究之餘仍然充滿輕鬆的氣氛，一起討論修課、研究內容幫助我能夠做得更快、更好，每次到實驗室總是有大家庭的感覺，讓我捨不得離開。

感謝我的家人與朋友，忙碌之餘就會一段時間沒有回家，但家人總是默默的支持我，不會給我太大的壓力。每次回家時，和家人與朋友相聚總是能釋放平時的壓力，讓我能夠繼續努力的度過研究生活。

doi:10.6342/NTU202001860

摘要

語音喚醒技術需要具有低耗能特性以利運行在計算資源限制的環境。換句話說，在低耗能限制下我們需要在精準度與延遲時間之間取得平衡。為了達成這些條件，端到端 (end-to-end) 模型比傳統大詞彙語音辨識 (large vocabulary continuous speech recognition, LVCSR) 方法更為合適，因為它使用較少記憶體。前人最好的方法深度殘差網路 (ResNets) 雖已達到很好的精準度，但模型仍然使用超過兩十萬個參數。為了解決這個問題，本篇論文提出以時延神經網路 (time-delay neural networks, TDNNs) 搭配對抗式訓練 (adversarial training) 之模型，使模型生成具有較少語者資訊的音素特徵，來達到好的精準度以及減少模型所需參數。本篇論文使用公開的資料集 Google Speech Commands 來訓練及衡量模型的表現。我們最好的模型使用一萬個參數 (達到深度殘差網路參數的 96% 減少) ，且錯誤率 (error rate) 4.3% 與其 4.2% 相距不大。除了參數量以外，運行時間也是一個重要的衡量標準，因此我們也將模型放入手機裝置來比較所有方法的表現，包含運行時間。基於在手機裝置上的測試，我們能夠決定最適合需求的模型。

關鍵字： 語音喚醒、時延神經網路、對抗式訓練、邊緣裝置

doi:10.6342/NTU202001860

Abstract

Small-footprint keyword spotting needs to use only small memory to run on computationally constrained environment. In other words, we need to strike a balance between accuracy and latency, under the constraint of small memory. To achieve this, end-to-end model is more suitable than Large Vocabulary Continuous Speech Recognition System (LVCSR) since it usually requires less memory. Previous state-of-the-art work based on ResNets achieved good accuracy on keyword spotting, but the model still used more than 200K parameters. To address this issue, this thesis presents a time delay neural networks (TDNNs) with adversarial training, which can generate phonetic features with less speaker information to achieve better accuracy and reduce number of model parameters. We used publicly available Google Speech Commands dataset to train and evaluate our models in this study. The best model of our study has 10K number of parameters (achieving 96% reductions of the ResNets model) with error rate 4.4%, which is comparable to the ResNets model's 4.2%. In addition to the number of parameters, latency is also an important metrics, so we put our models on a mobile device to compare all their performance, including latency. Based on this performance test on mobile phones, we can determine the model that suits our needs for various applications.

Keywords: Small-footprint Keyword Spotting, Time Delay Neural Networks, Adversarial Training, Edge Device

doi:10.6342/NTU202001860

Contents

doi:10.6342/NTU202001860

doi:10.6342/NTU202001860

doi:10.6342/NTU202001860

List of Figures

doi:10.6342/NTU202001860

doi:10.6342/NTU202001860

List of Tables

doi:10.6342/NTU202001860

Chapter 1

緒論

1.1 主題簡介

隨著人工智慧與機器學習的熱門發展，人們透過說話與智慧裝置互動變得相當常見，像是 Google 語音助理、siri、智慧音箱等。這些產品需要使用者透過觸碰設備或是語音喚醒來進行初始化，再進行對話、發出疑問、命令等。語音喚醒系統，透過偵測事先設定好的關鍵字 (keyword spotting, KWS) ，使智慧裝置透過持續的聆聽關鍵字來初始聲音輸入，讓使用者完全不需要使用雙手，對於某許多環境下是非常實用的，像是駕駛、遊玩 VR 設施等。

智慧裝置必須運行在計算資源限制的環境下，因此語音喚醒系統必須具有高精準度、低延遲、低耗能等特性。最初的方法透過大詞彙語音辨識 (large vocabulary continuous speech recognition, LVCSR) [7][8][9] 、關鍵字/填充字隱藏馬可夫模型 (keyword/filler hidden Markov model) [10] 來實現，但這些方法需要非常大的記憶體與計算量。近年來由於深度學習 [11] 的興起，深度關鍵字偵測網路 (Deep KWS) [1] 成為主流的模型，深度神經網路輸入聲學特徵後，輸出關鍵字的信心指數，當信心指數超過設定的閾值，則偵測到關鍵字。與 LVCSR、keyword/filler HMM 相比，深度神經網路實作更簡單，並且減少計算量以及記憶體的使用。目前最好的模型 residual networks (ResNets) [2] 達到了 4.2% 的錯誤率，但仍然使用到超過 20 萬的參數，對於低延遲、低耗能的要求，是相當不利的。

本篇論文希望改善參數的使用量，並且在錯誤率上也能有不錯的表現。我們提出以時延神經網路 (time-delay neural networks, TDNNs) [4] 搭配對抗式訓練

doi:10.6342/NTU202001860

(adversarial training) 的方法，抽出具有較少語者資訊的音素特徵，使模型不需要太過於複雜就能達到好的表現，達到使模型參數減少的目的。

此外過往的論文多僅展示參數與乘法數量，算是間接的衡量方法，因此我們實作在邊緣設備上，直接展示模型的實際效能、精準度，以及測量實際的延遲時間。

1.2 工具簡介

1.2.1 Pytorch

Pytorch[1]是一個基於 Torch 的 Python 開源機器學習程式庫，主要由 Facebook 的人工智慧小組開發，支持動態神經網路結構，同時能自動求導加速開發過程。對比 Tensorflow 的靜態神經網路結構，Pytorch 能夠任意改變神經網路的結構，且實現速度快，靈活度更勝於 Tensorflow。

1.2.2 Librosa

Librosa[2]是一個用於音訊分析、處理的 Python 工具包，用來進行資料的前處理如時頻轉換、特徵提取，再做為神經網路的輸入，有助於訓練的加速與結果的進步。本篇論文在端到端模型部分，利用 Librosa 進行特徵提取，Pytorch 進行神經網路的建構。透過深度神經網路的傳播，偵測輸入音訊是否為我們所設定的關鍵字。

1.2.3 Kaldi

Kaldi[3]是目前被廣泛使用的開源程式 [12]，由約翰·霍普金斯大學教授 Danial Povey 帶領的團隊，專門為了語音辨識來開發的一套軟體。Kaldi 功能涵蓋了整個語音辨識，從音訊中抽取特徵，HMM 的實作、DNN 架構建立、語言模型整合，也包含了模型訓練、獲取實驗數據、最後做成語音辨識的展示系統，所有程式碼都相當

[1]https://pytorch.org/
[2]https://librosa.github.io/librosa/index.html
[3]http://kaldi-asr.org/

doi:10.6342/NTU202001860

完善。另外 Kaldi 使用加權有限狀態轉換器 (weighted finite-state transducer, WFST) 來整合整個辨識系統，此方法最大的優點是非常快速，也很方便抽換系統裡任一個部分來做測試及比較。本篇論文在 DNN-HMM 模型部分，使用 Kaldi 來做語音辨識，偵測輸入音訊是否為我們所設定的關鍵字。

1.2.4 Pytorch Mobile

為了在設備上實現更高效的 ML，PyTorch 1.3 開始支持從 Python 到在 iOS 和 Android 上部署的端到端工作流，就是 Pytorch Mobile[4]。在邊緣設備上運行 ML 變得越來越重要，因為應用程式需要更低的延遲。它也是諸如聯合學習之類的隱私保護技術的基礎要素。

1.2.5 Vosk

Vosk[5]是 Alphacephei 公司所提供之離線語音辨識的 API，可以讓我們將 Kaldi 訓練完成的模型放到邊緣設備上運行。

1.3 章節概述

本篇論文分為四個章節:

第一章緒論，介紹本篇論文的研究主題以及使用的工具。

第二章文獻回顧，介紹前人的方法，找出能夠改進與突破的目標。

第三章研究方法介紹，詳細解釋語音喚醒技術，以及本篇論文所使用到的方法及理論。

第四章實驗，介紹使用的資料集、實驗流程的初始化與參數調整，以及實驗結果與錯誤分析。

第五章結論與未來展望。

[4]https://pytorch.org/mobile/home/
[5]https://alphacephei.com/en/

doi:10.6342/NTU202001860

Chapter 2

文獻回顧

2.1 關鍵字偵測深度網路

2.1.1 緒論

由 Google 於 2014 年提出 [1]，也是最早定義語音喚醒技術的論文。過往的關鍵字辨識，用於在句子中尋找特定的字，專注於有效率地在 lattice 中查找及索引出關鍵字的正確位置，lattice 往往相當大，解碼所需時間非常地長。語音喚醒技術則專注在快速無延遲地判斷使用者是否說出了事先定義好的關鍵字，HMM 的方法仍可以透過設計特別的 lattice 來實現，但瓶頸還是在於解碼所需要的時間。因此此篇論文提出深度神經網路用於語音喚醒，輸入通過模型後，再用後驗機率計算信心指數，即可預測是否為事先定義好的關鍵字，設計起來簡單、不需解碼能減少計算量，有效減少延遲時間。

2.1.2 方法介紹

關鍵字擷取主要分為三個步驟 (如 Figure 2.1 所示):

1. 特徵抽取: 抽取 PLP (perceptual linear prediction) 特徵作為模型輸入。
2. 神經網路預測: 模型需預測每個特徵為哪一個字，即輸出各個字的機率，以利後處理。

doi:10.6342/NTU202001860

3. 後處理: 利用模型輸出的機率，計算關鍵字的信心指數，若信心指數超過預先設定好的閾值，則偵測到關鍵字。

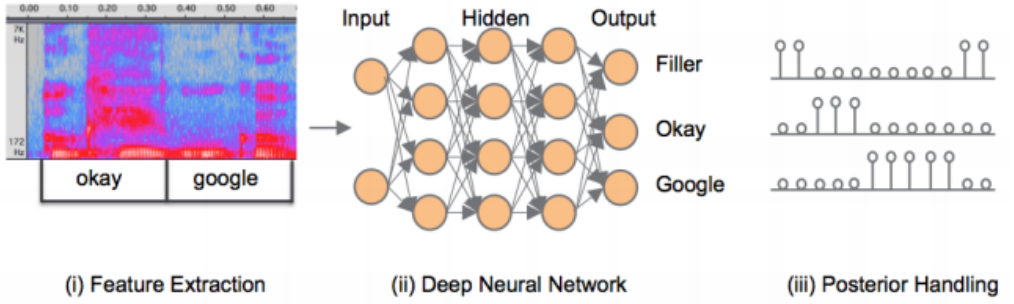


Figure 2.1: Deep KWS 架構 [1]

在後處理時，模型輸出的後驗機率帶有噪音，因此需要對特定大小窗格內的機率先做平滑化 (posterior smoothing)，令後驗機率 p_{ij} 為第 j 個特徵屬於第 i 個關鍵字的機率，$p_{ij}^{'}$ 為平滑化後的機率，窗格大小為 w_{smooth}，則公式為:

$$p_{ij}^{'} = \frac{1}{j - h_{smooth} + 1} \sum_{k=h_{smooth}}^{j} p_{ik} \tag{2.1}$$

其中 $h_{smooth} = \max\{1, j - w_{smooth} + 1\}$，即窗格內的第一個機率的編號。

再來計算信心指數 (confidence)，第 j 個特徵的信心指數需要從特定大小窗格內計算，令窗格大小為 w_{max}，則公式為:

$$confidence = \sqrt[n-1]{\prod_{i=1}^{n-1} \max_{h_{max} \le k \le j} p_{ik}^{'}} \tag{2.2}$$

其中 $h_{max} = \max\{1, j - w_{max} + 1\}$，即窗格內的第一個機率的編號。

最後透過調整不同的信心指數閾值，獲得 DET 曲線 (detection error tradeoff curve) 來比較深度網路與 HMM 方法的表現，X 軸為錯誤喚醒率 (false alarm rate, FAR)、Y 軸為錯誤拒絕率 (false reject rate, FRR)，因此曲線下面積越小越好。從 Figure 2.2 中可明顯看出，深度網路表現比 HMM 好很多，且在 0.5% FAR 時，深度網路之於 HMM 達到 45% 的改進。

doi:10.6342/NTU202001860

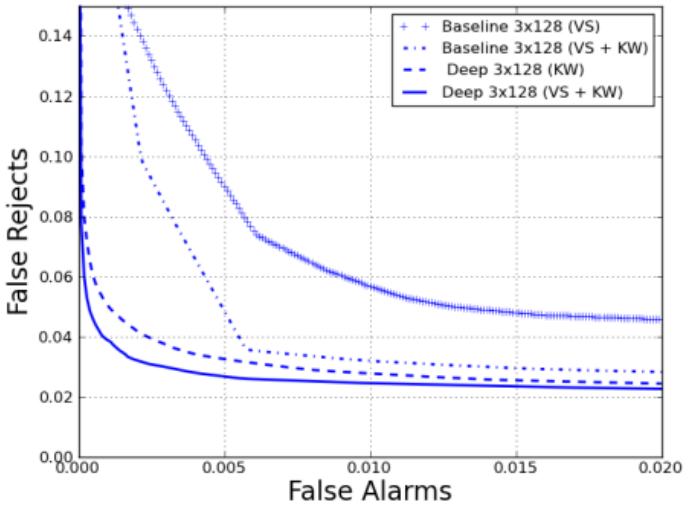


Figure 2.2: HMM vs. Deep KWS [1]

2.2 關鍵字偵測深度殘差網路

2.2.1 緒論

Google 在提出前面介紹的論文後，隔年使用了 convolutional neural networks (CNNs) 模型獲得了 27-44% 的進步 [13]，但代價是使用了上百萬的參數，也導致多層的卷積網路不易於訓練。深度殘差網路 (ResNets) 解決了深度卷積網路不易於訓練的問題，最初是用在影像辨識上，獲得了非常好的效果，而此篇論文 [2] 則將其運用到語音喚醒的模型中。

2.2.2 方法介紹

Residual 架構如 Figure 2.3 左半部所示，卷積網路的疊加會使訓練愈加困難，為了解決這個問題，模型在經過幾層卷積後，會將結果加上原始的輸入，能使模型變得比較好訓練。從數學上來看，我們令模型輸入為 x，正確的結果為 $H(x)$，模型的輸出為 $F(x)$，也就是說 $H(x) = F(x) + x$ 會比 $H(x) = F(x)$ 更好學習，尤其是在模型有非常深的架構時。而在每個卷積層後會有 ReLU 及 batch normalization。

完整模型架構如 Figure 2.3 右半部所示，第一層為不含 bias 的卷積層，加上 ReLU 及 batch normalization。之後則是一連串上述的 residual 架構，最好的模型使用了 6 個，加上隨著層數增加的 dilation (Figure 2.4)，可以增加模型每次卷積所看到的視野。最後通過 average pooling 及 softmax，輸出各個類別的機率。

doi:10.6342/NTU202001860

Model	Test accuracy	Par.	Mult.
tpool2	91.7% ± 0.344	1.09M	103M
res15	95.8% ± 0.484	238K	894M

Table 2.1: CNNs vs. ResNets

結果與比較列於 Table 2.1，從結果可以看出，透過 residual 的幫助，模型不需要疊的非常深就能有不錯的結果，反映在參數的總數上，約只有 CNNs 的五分之一，而且精準度獲得了大量的改進。

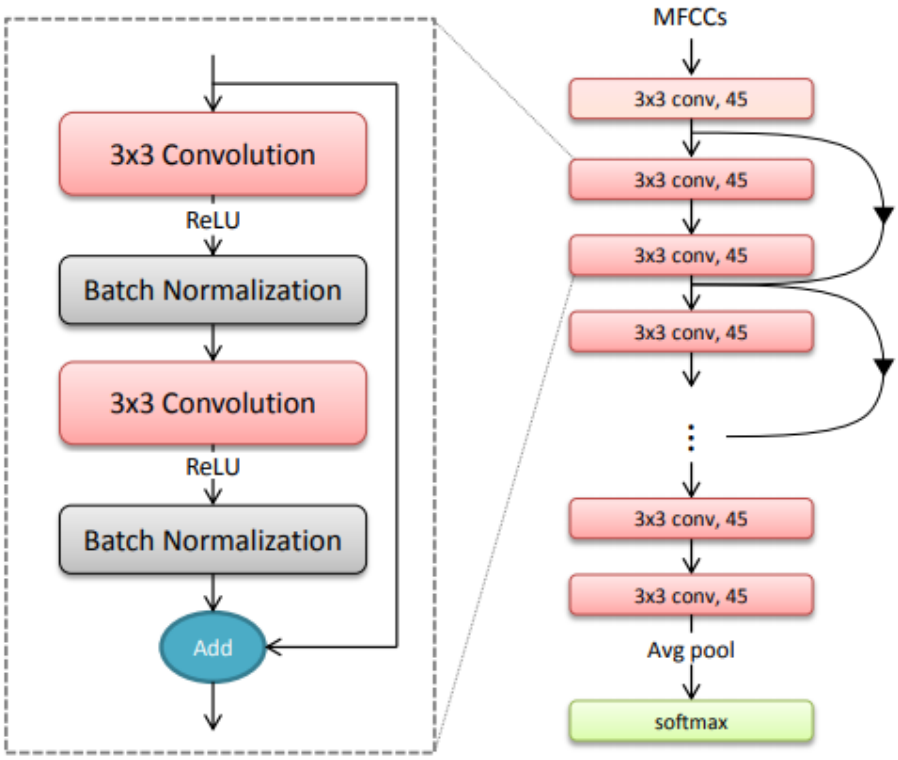


Figure 2.3: ResNets 模型架構圖 [2]

2.3 關鍵字偵測時延神經網路搭配權重共享自注意機制

2.3.1 緒論

前面提到的 ResNets 獲得了很好的表現，但是使用了上百萬的參數，為了解決這個問題，此篇論文 [3] 提出以時延神經網路 (time delay neural network, TDNNs) 搭配權重共享自注意機制 (shared weight self-attention, SWSA)，自注意機制 (self-attention) [14] 不管在影像、語音領域都帶來很大的突破，此處透過權重共享減

doi:10.6342/NTU202001860

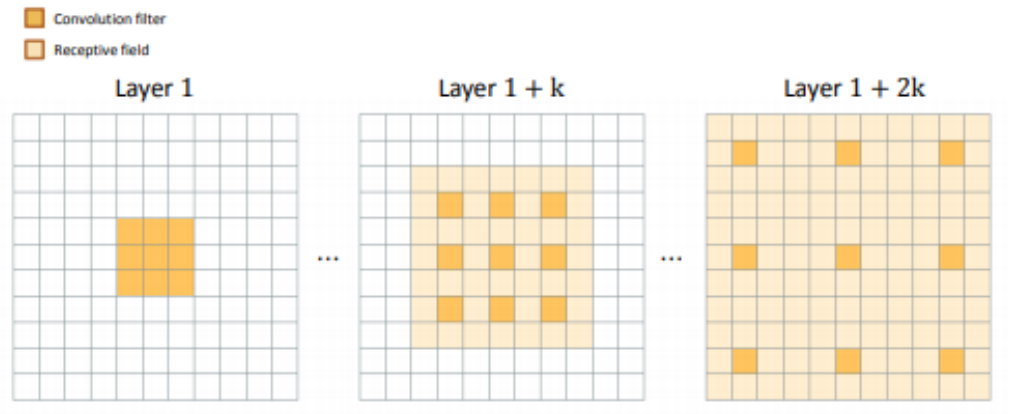


Figure 2.4: Dilation convolution [2]

少參數的使用，並且不會使表現有太大的減少。模型參數 (12K) 達到了 ResNets (238K) 的 1/20，並且錯誤率是可與之相比的 4.19%。

2.3.2 方法介紹

模型架構如 Figure 2.5 所示，第一層為 TDNN sub-sampling，將輸入的特徵長度縮小，第二層為此篇論文所提出的 SWSA 層，第三層與第四層為單純的 TDNN，最後將所有音框進行平均，通過 softmax 層後分為 11 類，分別為 10 個關鍵字及 Filler 代表所有的非關鍵字。

受到 self-attention 在很多任務成功的啟發 [14][15][16]，此篇論文提出 SWSA 來減少參數量，原理圖如 Figure 2.5 中的虛線框所示，輸入音訊透過 SWSA 獲取長期依賴 (long term dependencies) 後，再通過 ReLU 以及 layer normalization。

原始的 self-attention 可以寫成:

$$\begin{aligned} &\text{Attend}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{D_k}})V, \\ &\text{where } Q = UW_q, K = UW_k, V = UW_v \end{aligned} \tag{2.3}$$

$Q \in \mathbb{R}^{T_u \times D_k}$、$K \in \mathbb{R}^{T_u \times D_k}$、$V \in \mathbb{R}^{T_u \times D_v}$ 分別為 query、key 及 value，$U \in \mathbb{R}^{T_u \times D_u}$ 為輸入的矩陣。T_u 是輸入矩陣 U 的長度，D_k 是 Q、K 的維度，D_v 是 V 的維度。W_q、W_k、W_v 則將 U 投影到不同的空間。

雖然 self-attention 展現了很好的表現，但三個投影矩陣 W_q、W_k、W_v 需要使用較多的參數量。因為 self-attention 的核心計算部分為內積運算，我們考慮三個

doi:10.6342/NTU202001860

Model	Error Rate	#Para.
res15	4.2%	238K
tdnn-swsa	4.19%	12K

Table 2.2: ResNets vs. tdnn-swsa

矩陣必須投影到同一個空間，如此我們就能將 SWSA 寫成:

$$\begin{aligned} &\text{Attend}(V) = \text{softmax}(\frac{VV^T}{\sqrt{D}})V, \\ &\text{where } V = UW \end{aligned} \tag{2.4}$$

$W \in \mathbb{R}^{D_u \times D}$ 為共享權重的矩陣將輸入投影至 value。接著就能將 attention 運算在 value 上，因此所使用到的投影矩陣由三個減少為一個，Figure 2.6 展示了 SWSA 的運算。

SWSA 也可以像 self-attention 一樣延伸成 multi-head 的版本，可以寫成:

$$\begin{aligned} &\text{Multihead}(V) = \text{Cat}(h_1, ..., h_n), \\ &h_i = \text{Attend}(UW_i) \end{aligned} \tag{2.5}$$

其中 n 為 head 的數量，$W_i \in \mathbb{R}^{D_u \times (D/n)}$ 為每個 head 的投影矩陣，D/n 必須為整數，所以 head 的個數不會影響 SWSA 的參數量，最後再將所有的 head 連接起來即可。

結果與比較列於 Table 2.2，透過 SWSA 使模型參數有大幅度的減小，且結果甚至比 ResNets 還好了 0.01%，是現在最好的方法。

doi:10.6342/NTU202001860

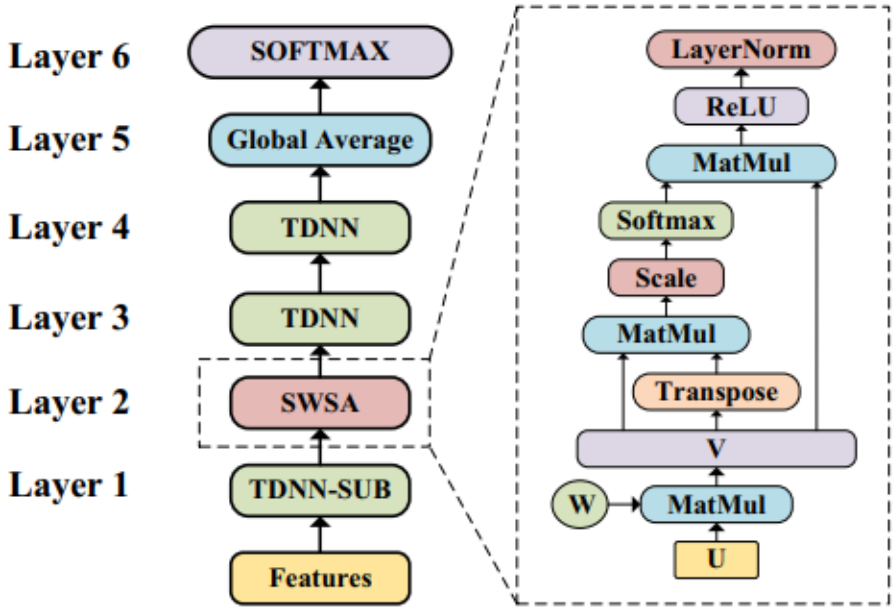


Figure 2.5: TDNN-SWSA 模型架構圖 [3]

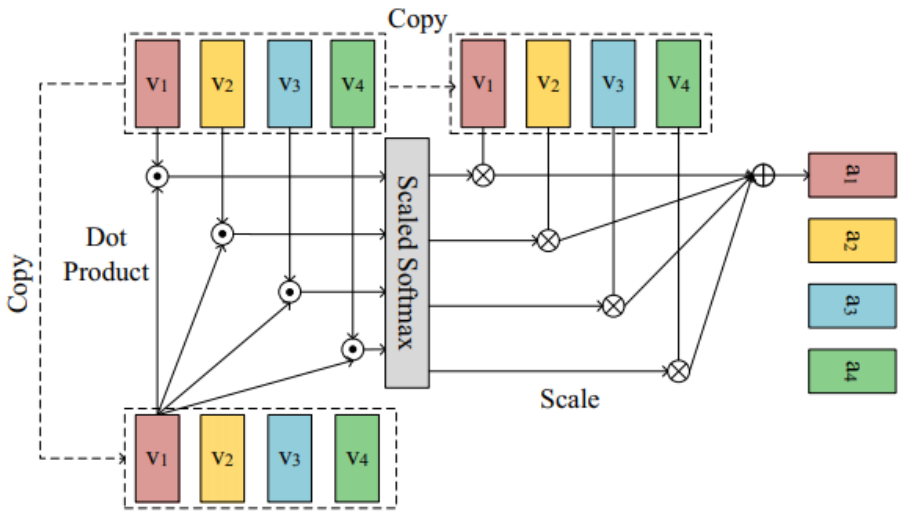


Figure 2.6: SWSA 計算方式 [3]

doi:10.6342/NTU202001860

Chapter 3

研究方法

3.1 問題定義

語音喚醒系統運作在智慧裝置，這些裝置往往是計算資源有限的，因此所使用的方法必須具備低耗能、低計算量、低記憶體使用等特性，使延遲時間不至於過長，而關鍵字偵測是實現語音喚醒技術的方法。所謂的關鍵字偵測，透過事先設定好的關鍵字來進行模型的訓練，如 Google 的 “Okay, Google“ 、Apple 的 “Hey, Siri“ 等，模型輸入音訊後，輸出關鍵字的信心指數，若信心指數超過事先定義好的閾值，則偵測為關鍵字，反之則為填充字。閾值的選擇影響著錯誤喚醒率、錯誤拒絕率，用來衡量模型的表現。

端到端模型的輸入有固定的長度，本實驗使用的是一秒鐘，實際應用時，裝置會持續聆聽音訊，將一秒音檔輸入進模型進行預測後，馬上將音訊平移 0.01 秒再輸入進模型進行預測，如此不斷地重複進行。由於模型預測時間往往低於 0.01 秒，因此能夠即時的判定是否能夠喚醒裝置。

3.2 特徵抽取

3.2.1 梅爾頻率倒譜係數

[1]http://speech.ee.ntu.edu.tw/DSP2018Autumn/Slides/7.0.pptx

doi:10.6342/NTU202001860

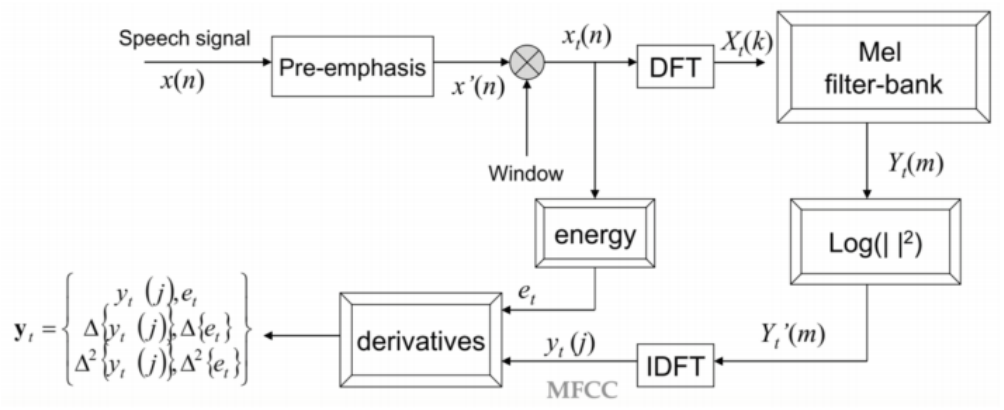


Figure 3.1: MFCC 流程圖[1]

梅爾頻率倒譜係數 (Mel-frequency cepstral coefficients，MFCCs) 由 Davis 和 Mermelstein 於 1980 年代提出，成為語音辨識常用的聲學特徵，為組成梅爾頻率倒譜的係數。它衍生自音訊片段的倒頻譜 (cepstrum)，梅爾頻率倒譜的頻帶劃分是在梅爾刻度上等距劃分，相比於用正常的對數倒頻譜中的線性間隔，梅爾頻帶更能近似人類的聽覺系統。透過仿造人類聽覺，機器更能夠從中學習，在語音辨識獲得了更好的表現。Figure 3.1 為抽取 MFCC 的標準流程。

主要分為七個步驟:

1. 將語音訊號預強化 (pre-emphasis)，即通過一個高通濾波器。
2. 將一段語音訊號分解為多個音框 (frame)。
3. 進行傅立葉轉換，將訊號轉換至頻域。
4. 將頻譜 (spectrum) 通過梅爾濾波器 (三角重疊窗口)，得到梅爾刻度。
5. 在每個梅爾刻度上提取對數能量。
6. 對上面獲得的結果進行離散傅立葉反轉換，轉換到倒頻譜域。MFCC 為此倒頻譜圖的幅度 (amplitudes)。一般使用 12 個係數，與音框能量疊加得 13 維的係數。

MFCC 特徵抽取流程的第一步驟為預強化 (pre-emphasis)，目的是為了消除發聲過程中聲帶和嘴唇的效應，來補償語音訊號受到發音系統所壓抑的高頻部分 (另一種說法則是要凸顯在高頻的共振峰)。人體發出的聲音可分為兩種類型，一種

doi:10.6342/NTU202001860

是在時域上具有明顯基本週期的有聲音 (voiced sound)，另一種則是不具有明顯週期的無聲音 (unvoiced sound)，其中有聲音對於語音訊號的分析較能提供有用的資訊，但有聲音的高頻部分容易在傳輸過程中損耗，導致有聲音呈現低頻強、高頻弱的強度分佈。這樣的衰減趨勢導致一些重要語音資訊流失，例如共振峰結構 (formant structure) 變得不明顯，預強化即用來彌補這些流失資訊的方法。預強化方法為將語音訊號通過一個高通濾波器，表示為:

$$H(z) = 1 - a \cdot z^{-1} \tag{3.1}$$

假設原始聲音訊號在時域上以 $s(n)$ 表示，則通過預強化後的新訊號 $s^{'}(n)$ 可表示為:

$$s^{'}(n) = s(n) - a \cdot s(n-1) \tag{3.2}$$

其中 a 通常介於 0.9 和 1.0 之間。

其原理為假設原始訊號 $s(n)$ 經 Z 轉換 (Z-transform) 後可表示為:

$$Z[s(n)] = X(z) \tag{3.3}$$

預強化後的訊號經 Z 轉換後則為:

$$Z[s(n) - a \cdot s(n-1)] = X(z) \cdot (1 - a \cdot z^{-1}) \tag{3.4}$$

因此，在頻域上將原始訊號乘上 $(1 - a \cdot z^{-1})$ 即為預強化後的訊號，此值於低頻時接近 0，越往高頻靠近則越接近 1，可補償語音訊號中衰減的高頻部分。

MFCC 特徵抽取流程的第二步驟為將訊號分解為多個音框 (frame)，因為聲音訊號會隨著時間不停的改變它們的特性，只有在短時間間隔內才能保持特性的不變。

MFCC 特徵抽取流程的第三步驟為離散傅立葉轉換 (discrete fourier transform, DFT)，由於訊號在時域上的變化通常很難看出特性，所以將訊號從時域轉換到頻域上來觀察能量分布，轉換過後的訊號稱為頻譜 (spectrum)。

[2]http://speech.ee.ntu.edu.tw/DSP2018Autumn/Slides/7.0.pptx

doi:10.6342/NTU202001860

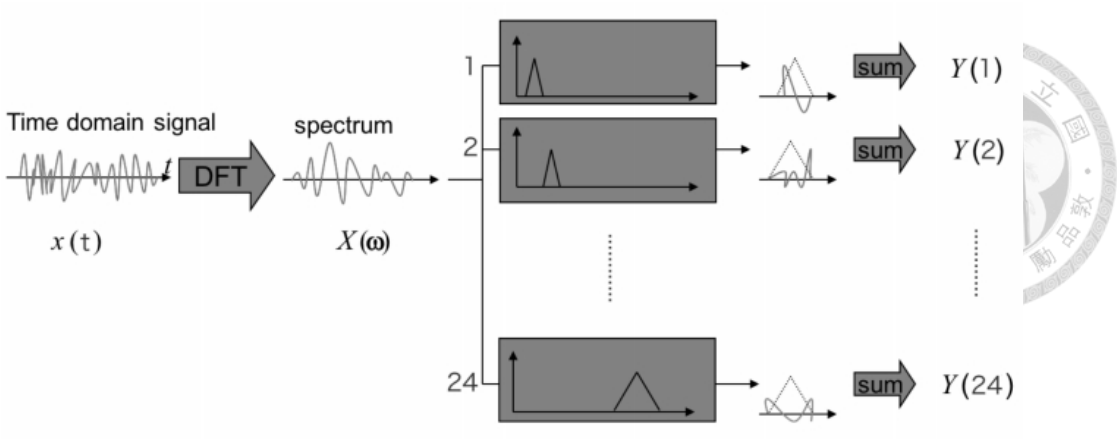


Figure 3.2: 離散傅立葉變換與梅爾濾波器[2]

MFCC 特徵抽取流程的第四步驟為將頻譜通過梅爾濾波器群 (Mel-filter bank) ，獲得各自與頻率之係數和，如 Figure 3.2。梅爾濾波器群為 24 個三角帶通濾波器，各濾波器對應之頻帶則稱為關鍵頻帶 (critical band) 。這些關鍵頻帶是依據人類兩大內耳特性所設計:

1. 人類的內耳構造為螺旋狀，其內部的不同區段各自負責不同的音訊頻帶之處理，相鄰的區段所負責的頻帶是會重疊的，如 Figure 3.3，因此相鄰的關鍵頻帶在設計上也是會有所重疊。

2. 內耳越靠近外部的區段所負責的頻帶越高，而且頻率的增長速度不一，較靠近內部的區段 (約 200Hz 至 1000Hz) ，頻率增長速率近似等速，但較靠近外部的區段 (約 1000Hz 之後) ，則近似於指數增長，如 Figure 3.3，這代表人類的內耳對高頻率的聲音解析度較差，意即對高頻段的聲音較不敏感。

MFCC 特徵抽取流程的第五步驟為取對數能量，使用前一步驟所獲得的 24 個係數取絕對值與平方，再進行對數運算。取對數有幾點原因:

1. 人類聽覺對於訊號在對數是成比例的。

2. 對數運算壓縮大的值、擴大小的值，與人類聽覺系統特性相同。

3. 能夠使特徵抽取對於動態訊號變化較不敏感。

MFCC 特徵抽取流程的最後步驟為取離散傅立葉反轉換 (inverse discrete fourier transform, IDFT) ，將前一步驟得到的 24 個係數做 IDFT，但只保留前面 12 個係

doi:10.6342/NTU202001860

數，因為丟棄高倒頻域值，代表一個類似低濾波器的概念，可以使訊號平滑化，增進語音處理的性能。除此之外，通常還會再額外加上另一個代表能量的係數，假設訊號在一個音框內表示為 $x(t)$，則一般常見之能量係數 e 之計算方式為:

$$e = 10 \cdot \log(\sum_{t} x(t)^2) \tag{3.5}$$

則可獲得 13 維的梅爾頻譜倒譜係數 (Mel-frequency cepstral coefficients, MFCCs) 。通常 MFCC 在使用時，會再計算一階與二階倒數，加上原本的 13 維合成 39 維的 MFCC。

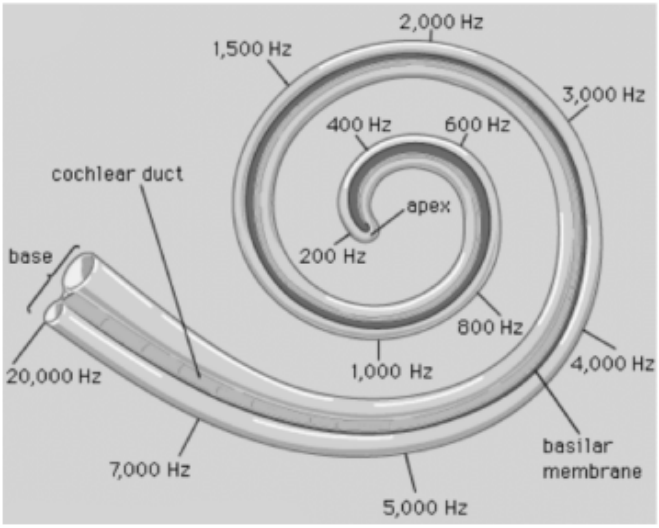


Figure 3.3: 人類的內耳構造[3]

3.3 網路層架構

3.3.1 時延神經網路

時延神經網路 (time-delay neural networks, TDNNs) 為近期語音辨識常使用的神經網路架構，目的是為了提取更多音框的前後文資訊，提升 DNN 架構的 senone 分類的準確率，進而提升語音辨識的效果。

TDNN 於 1989 年提出 [4]，目的是做音素辨識 (phoneme recognition) 。如 Figure 3.4 所示，目的是預測輸入的 15 個音框屬於「B」、「D」、「G」中哪一個音

[3]http://speech.ee.ntu.edu.tw/DSP2018Autumn/Slides/7.0.pptx

doi:10.6342/NTU202001860

素，輸入為 15 個 16 維的音框。在第一層網路中同時考慮三個音框 (t、t+1、t+2)，稱之為濾波器，使用 8 個濾波器後第二層即獲得 8 維的特徵。同時每次移動一個音框，能再獲得下一個時間點透過濾波器所得到的特徵。同時考慮多個音框能有許多好處：音框的前後通常是有關係的，可以學習音框間的變化；若濾波器只考慮一個音框 (即 DNN) ，第一個音框與最後一個音框雖然都屬於同一個音素，但兩著仍會存在差異，此時濾波器相同便不太合理。這樣的架構如同一維的卷積神經網路 (convolutional neural network, CNN) 。透過堆疊多層的 TDNN，在最後的輸出層時，模型其實已經看完了所有的音框，也就包含了音框前後文的資訊。因此這個架構同時具備獲取局部與全局資訊的能力，能使模型學習到更好的分類效果。

前面的例子中只有 15 個音框，但語音資料往往擁有上百上千個音框，加上模型需要層層推疊的情況下，所使用到的參數量及計算量會非常的龐大，但其實上層的神經元計算了許多重複的音框。為了解決這個問題，sub-sampling 技術於 2015 年被提出 [5]。如 Figure 3.5 所示，對於輸出 t 來說，包含了 $(t-13)$ 到 $(t+9)$ 共 23 個音框的資訊，沒有 sub-sampling 的狀況下，計算了所有藍色加紅色的線，但在只計算紅色線的狀況下，可以看到仍然計算了所有輸入的音框，與原本相比，減少了非常多的計算量以及參數量。此方法可以降低運算複雜度、使訓練速度大幅提升。

TDNN 能夠感知輸入資料時間性的關係，這個特性與遞歸神經網路 (recurrent neural network, RNN) 類似，差別在於 DNN 及 TDNN 的每個隱藏層只會接收前一個隱藏層的輸出，而 RNN 中的隱藏層除了接收前一個隱藏層的輸出外，還會將自己前一個時間點的輸出也當成輸入。這樣的結構使得 RNN 在理論上可以學習到無限長度的歷史信息，但是在訓練過程中，也會面臨著複雜度較高以及梯度消失的問題，因此 TDNN 對於語音辨識來說是較為適合的架構。

3.3.2 卷積神經網路

卷積神經網路 (convolutional neural network, CNN) [17] 主要應用在電腦視覺 (computer vision, CV) 領域，其架構如 Figure 3.6 所示。在卷積神經網路中，每層網路皆有多個核心 (kernel) ，每個核心會與輸入的圖片進行卷積運算，卷積運算即是將核心中的所有參數與圖片中對應到的區域中所有像素 (pixel) 相乘並相加得到該

doi:10.6342/NTU202001860

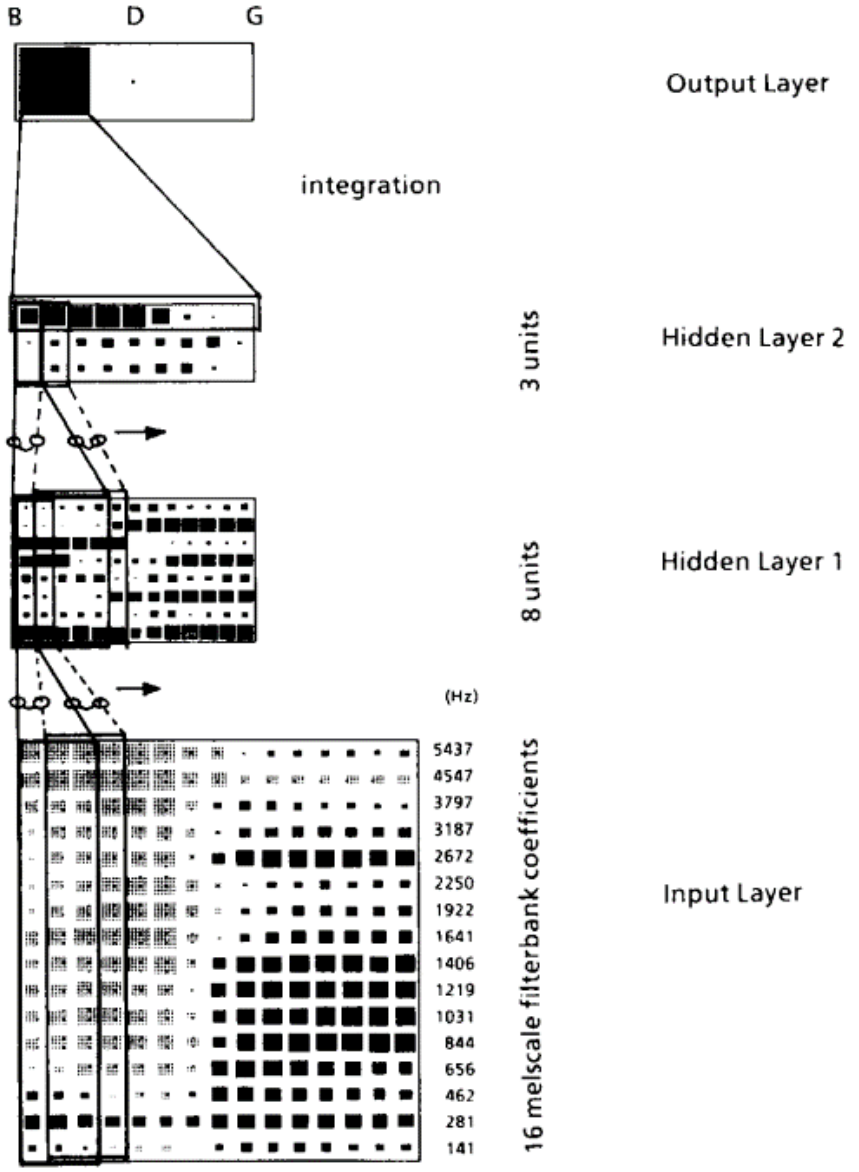


Figure 3.4: 時延神經網路架構 [4]

區域的特徵值如 Figure 3.7 所示，接著移動核心使之對應到下個區域計算下個區域的特徵值，當該核心掃視完整張圖片後便會得到一張特徵圖 (feature map) ，接著再以下一個核心對同一張圖抽取特徵。在卷積神經網路中，每一層當中可能會有數百個核心，皆擁有不同的參數，分別用來抽取輸入圖片的不同特徵。在語音領域的研究中，卷積神經網路也得到了很大的重視，研究人員將輸入音訊透過傅利葉轉換轉成時頻圖後輸入到網路中，往往能得到較僅用全連接網路或循環神經網路更佳的結果。

[4]https://mc.ai/convolution-operation-comprehensive-guide/

doi:10.6342/NTU202001860

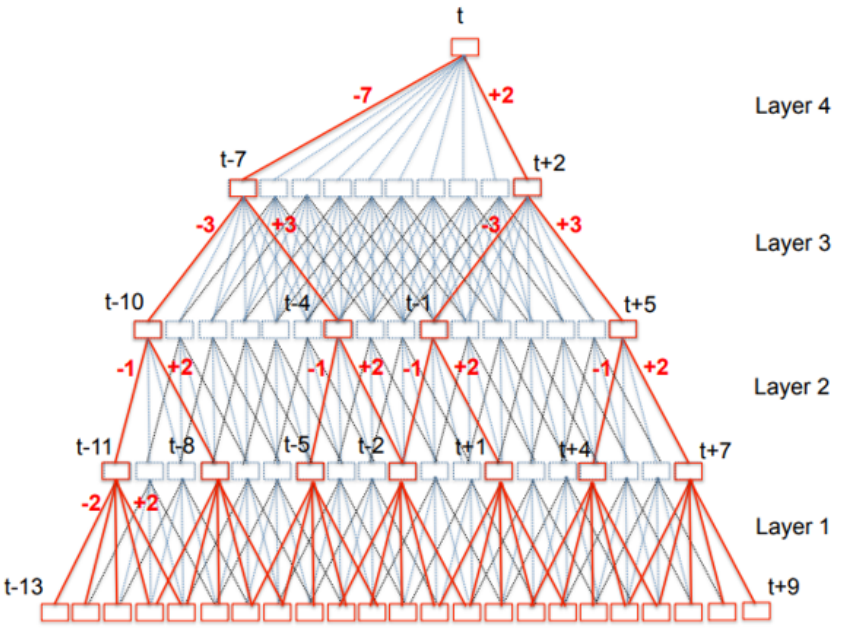


Figure 3.5: 時延神經網路架構 sub-sampling 技術 [5]

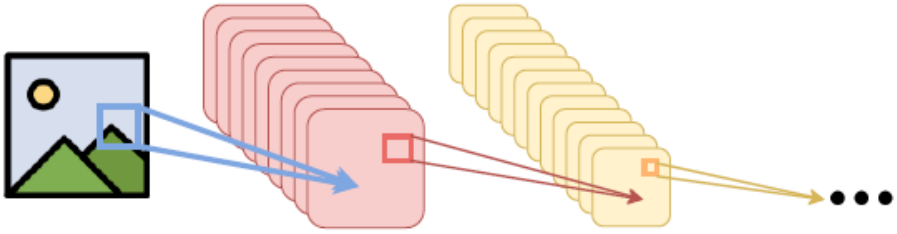


Figure 3.6: 卷積神經網路示意圖

3.3.3 循環神經網路

循環神經網路 (recurrent neural network, RNN) [18] 主要應用在自然語言處理 (natural language processing, NLP) 、數位語音處理 (digital speech processing, DSP) 等領域，其架構如 Figure 3.8 所示。在循環神經網路中，在每個時間點模型皆有兩個輸入，除了當時的輸入外，還會輸入上一個時間點的輸出，由於這個特性讓循環神經網路具有記憶的功能，因此對具有時序性質的資料能有較佳的結果。但是循環神經網路存在著記憶時間過短的問題，這個問題在長短期記憶當中得到解決。

長短期記憶 (long short-term memory, LSTM) [19] 主要架構與循環神經網路相同，但其解決了循神經網路中記憶範圍過短的問題。解決方法在於每層網路的設計，如 Figure 3.9 所示，在長短期記憶中，每個時間點的輸入在進入神經元 (neuron) 前，皆有輸入閘門 (input gate) 來控制目前輸入是否進入到神經元的記憶

doi:10.6342/NTU202001860

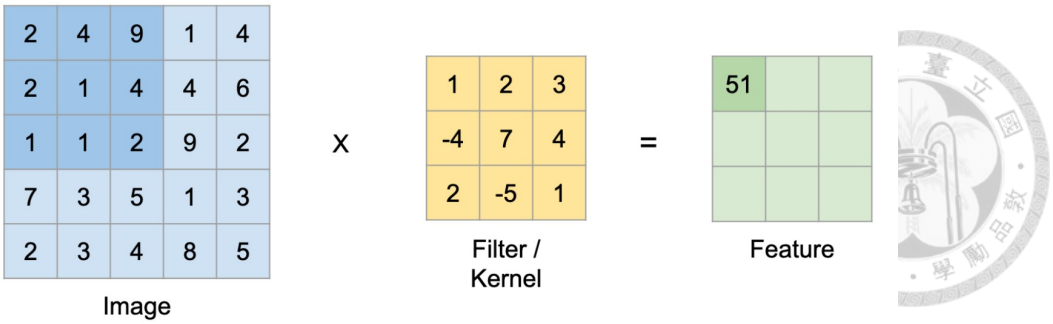


Figure 3.7: 卷積運算示意圖[4]

中，並且有個遺忘閘門 (forget gate) 控制上一個階段記憶哪些需要保留哪些需要遺忘，最後有個輸出閘門 (output gate) 控制神經元的輸出。與一般循環神經網路最大的不同在於，循環神經網路每個時間點一定要輸入上個時間點的輸出，這會導致網路在經過多個時間後會忘記前面的資訊，長短期記憶在加入遺忘閘門的設計後，可以選擇記住較重要的資訊，並將不重要的資訊拋棄，因此對長時序的資料表現較佳。

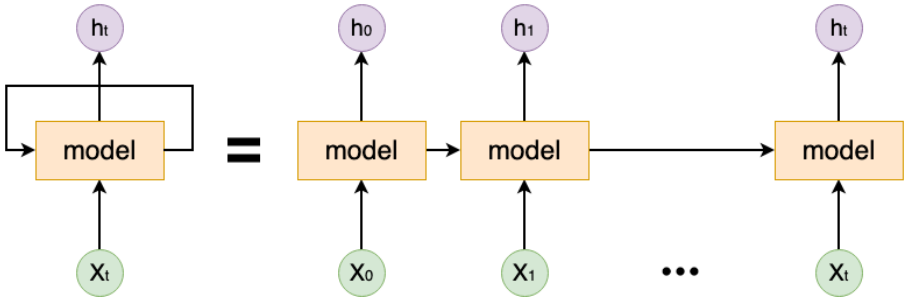


Figure 3.8: 循環神經網路示意圖

3.4 對抗式訓練

在語音特徵中，其實包含了很多的資訊，像是文字、語者、抑揚頓挫等，每個資訊都會影響著模型預測的結果，因此在特徵的選擇上，自古以來就是非常重要的問題，對於語音辨識領域來講，若能得到只有文字資訊的特徵來當作模型的輸入，概念上會提高模型預測的表現；反之在語者辨識的領域中，若能得到只有語者資訊的特徵來當做模型的輸入，也能得到更好的表現。由此想法出發則有了

[5]https://zh.wikipedia.org/wiki/長短期記憶

doi:10.6342/NTU202001860

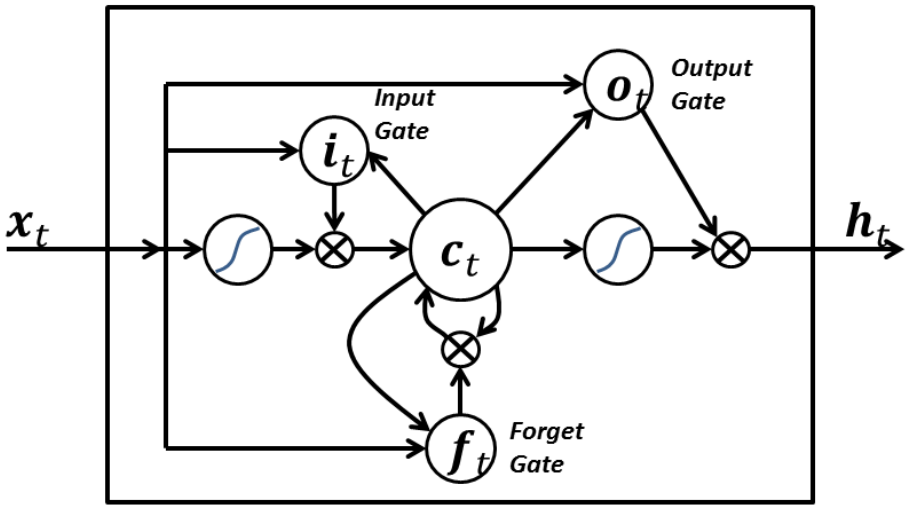


Figure 3.9: 長短期記憶示意圖[5]

feature disentanglement 的技術，此技術即為將特徵中某些資訊去除，只留下我們需要的資訊。

對抗式訓練 (adversarial training) 為其中一個能夠實現此技術的方法，最早被用來解決訓練及測試時資料所在域 (domain) 的不同所導致結果不佳的問題，稱為域對抗式訓練 (domain adversarial training, DAT) [6]。現在廣為人知的生成對抗網路 (generative adversarial network, GAN) [20] 中也有對抗式訓練的存在。此處我們透過將文字資訊與語者資訊分為兩個訓練目標，透過對抗的方式使模型學習我們需要的特徵抽取，也就是學習文字資訊同時捨棄語者資訊，如此一來就能獲得 Feature Disentanglement 的效果。

模型架構如 Figure 3.10 所示，綠色的部分稱為特徵抽取器，也就是模型需要學習到只抽取文字特徵的部分；藍色的部分則是文字分類器，模型需要學習正確的分類各個關鍵字；紅色的部分則是語者分類器，模型需要學習正確的分類各個語者。對於文字分類器及語者分類器而言，目標都是正確的分類，因此就是我們熟知的計算 loss、反向傳播、梯度下降，這兩個分類器就能學習到各自的目標。但對於特徵抽取器來說，我們可以從圖中看到，他會接收來自兩個分類器的 loss，特徵抽取器的目標是要學習抽取文字特徵、捨棄語者特徵，因此來自語者分類器的 loss 必須翻轉 (gradient reversal) ，來自文字分類器的 loss 則不變，透過 loss 對抗的概念，使特徵抽取器學習到留下文字特徵、捨棄語者特徵。

現在讓我們用一般表示法來說明 DAT 中不同部分的運作。令 $G_f(\cdot;\theta_f)$ 為參數

doi:10.6342/NTU202001860

是 θ_f 的特徵抽取器；令 $G_y(\cdot;\theta_y)$ 為參數是 θ_y 的文字分類器；令 $G_d(\cdot;\theta_d)$ 為參數是 θ_d 的語者分類器。

我們可以將文字分類 loss 與語者分類 loss 分別表示為:

$$\mathcal{L}_y^i(\theta_f,\theta_y) = L_y(G_y(G_f(x_i;\theta_f);\theta_y), y_i) \tag{3.6}$$

$$\mathcal{L}_d^i(\theta_f,\theta_d) = L_d(G_d(G_f(x_i;\theta_f);\theta_d), d_i) \tag{3.7}$$

接著我們進行參數的更新，兩個分類器需要最小化各自的 loss，而特徵抽取器要最小化文字特徵的 loss、最大化語者特徵的 loss，因此三個分類器的梯度下降分別表示為:

$$\theta_f \leftarrow \theta_f - \mu(\frac{\partial \mathcal{L}_y^i}{\partial \theta_f} - \lambda\frac{\partial \mathcal{L}_d^i}{\partial \theta_f}) \tag{3.8}$$

$$\theta_y \leftarrow \theta_y - \mu\frac{\partial \mathcal{L}_y^i}{\partial \theta_y} \tag{3.9}$$

$$\theta_d \leftarrow \theta_d - \mu\lambda\frac{\partial \mathcal{L}_d^i}{\partial \theta_d} \tag{3.10}$$

其中 μ 是學習率 (learning rate)，λ 是超參數 (hyperparameter)，兩個分類器反覆地訓練。

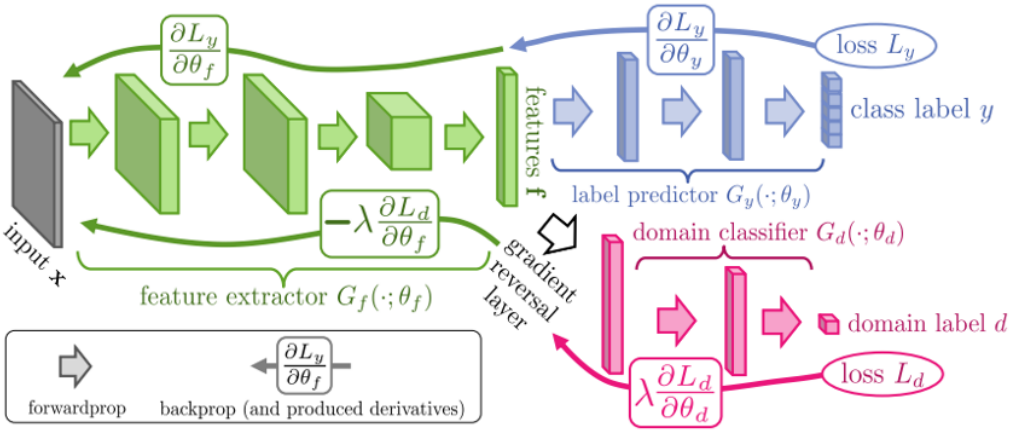


Figure 3.10: 域對抗式網路架構 [6]

doi:10.6342/NTU202001860

Chapter 4

實驗

4.1 資料集介紹

Google Speech Commands 是一個公開的資料集，由 Google 於 2017 年 8 月釋出 [21]。裡面一共有 64,727 個一秒鐘左右的音檔，sample rate 皆為 16,000，共來自 1,881 個語者，且含有 30 個詞列於 Table 4.1。

本篇論文延用資料集內的原始設定，將 10 個詞當成 keywords，剩下 20 個詞則當成 fillers 如 Figure 4.1 所示。此外，分配訓練集 51,088 個音檔、驗證集 6,798 個音檔、測試集 6,835 個音檔，各為 80%、10%、10%，也延用資料集內的設定，以利之後與其他方法做比較。

此外，為了實現本篇論文所提出的對抗式訓練方法，我們從檔名中提取了每個音檔的語者標籤，訓練、驗證、測試的語者互不重複。

Keywords	“down“, “go“, “left“, “no“, “off“, “on“, “right“, “stop“, “up“, “yes“
Fillers	“bed“, “bird“, “cat“, “dog“, “happy“, “house“, “marvin“, “sheila“, “tree“, “wow“, “zero“, “one“, “two“, “three“, “four“, “five“, “six“, “seven“, “eight“, “nine“

Table 4.1: Google Speech Commands dataset

doi:10.6342/NTU202001860

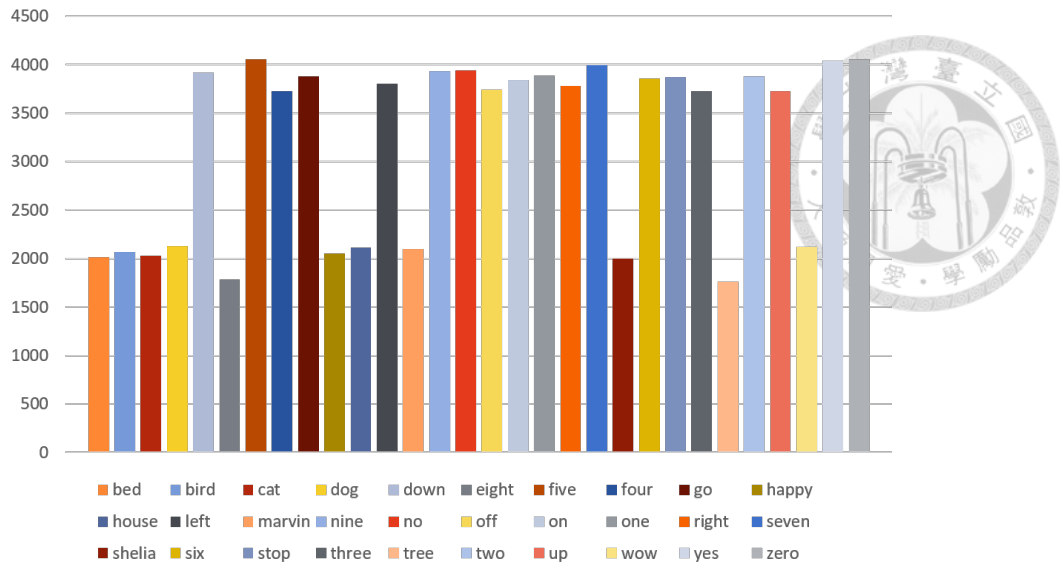


Figure 4.1: 每個音檔數量

4.2 實驗流程

4.2.1 聲學特徵抽取

DNN-HMM 部分，實驗使用的聲學特徵為 MFCC，音檔的取樣率 (sample frequency) 為 16000，音框大小為 400，音框每次位移長度為 160，因此每秒鐘共有 101 個音框，也就是有 101 個 MFCC 特徵。特徵選用 40 維高解析度 MFCC (40-dimensional high resolution MFCC)，對 40 個 Mel-frequency bins 做離散餘弦轉換 (discrete cosine transform, DCT) 後，再做 CMVN (cepstral mean and variance normalization)。作為 TDNN 的輸入時，每個 kernel 會再串聯 30 維的 i-vector，假設 kernel 大小為 N，則輸入為 $40N + 30$ 維的聲學特徵 [22]。

端到端部分，實驗使用的聲學特徵為 MFCC，音檔的取樣率 (sample frequency) 為 16000，音框大小為 512，音框每次位移長度為 128，因此每秒鐘共有 126 個音框，也就是有 126 個 MFCC 特徵。特徵選用無正規化 40 維高解析度 MFCC (un-normalized 40-dimensional high resolution MFCC)，此為 google 使用在 DNN 模型的參數，由 40 個 Mel-frequency bins 得來，與過去 13 維加上一階及二階倒數串成的 39 維 MFCC 略有不同。

doi:10.6342/NTU202001860

4.2.2 神經網路架構

DNN-HMM 部分使用 STL-TDNN 模型 (Figure 4.2)，聲學特徵經過 5 層 TDNN layer，最後一層為全連接層 (fully-connected layer)，每一層的有 512 個神經元，最後輸出 triphone-state (senome)，Figure 4.2 中每個 TDNN layer 括號中的數字表示 layer-wise context 資訊，0 表示當下時間點 t，-1 為 (t-1)，1 為 (t+1)，除了第一層 TDNN layer 將 (t-2) 到 (t+2) 所有輸入特徵向量納入計算外，其餘的皆為 sub-sampling，這樣的隱藏層架構設定是由 Kaldi 中 WSJ 的範本提供的腳本。除此之外，語言模型部分會特殊設計成適合辨識關鍵字的架構，如 Figure 4.3 所示。

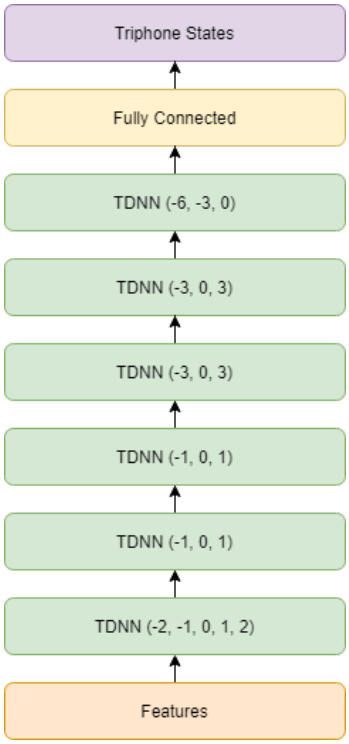


Figure 4.2: STL-TDNN 架構圖

端到端部分我們會實驗多種不同的架構，第一個架構為多層的 TDNN，目的是為了比較有無對抗式訓練所帶來的差異。接著在我們的對抗式網路中，藉由替換不同的特徵抽取層來查看各自的效果。

純 TDNN 模型架構圖如 Figure 4.4 所示，輸入為無正規化 40 維高解析度 MFCC，第一層用 sub-sampling 將輸入特徵長度縮小，可以使模型的計算量減少，再經過兩層一般的 TDNN 層，此時輸出的音框數為原本的三分之一再減四，因此需要將所有音框進行平均，獲得一個飽含資訊的音框，最後再經過 softmax 層幫

doi:10.6342/NTU202001860

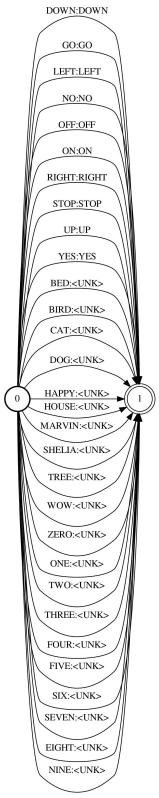


Figure 4.3: 語言模型

助我們進行關鍵字的分類。

接著我們由純 TDNN 模型出發，讓原本的模型利用對抗式訓練使結果更好，架構圖如 Figure 4.5 所示，首先第一層我們稱為特徵抽取器 (encoder)，輸入為無正規化 40 維高解析度 MFCC，輸出稱為隱藏特徵，隱藏特徵會送到兩個不同的分類器，分別為關鍵字分類器 (speech classifier) 與語者分類器 (speaker classifier)，關鍵字分類器的功能與原本相同，即進行關鍵字的分類；而新增加的語者分類器，則進行語者的分類。一般的模型訓練時，會先進行前向傳播，獲得結果後計算 loss，之後進行反向傳播，藉由參數的更新使模型學習得更好。在對抗式訓練時，關鍵字分類器與語者分類器皆為一般的更新方式，但特徵抽取器的參數更新，有稍微不同。我們可以從圖中看到，對於關鍵字分類的 loss，是一般的反向傳播，使參數更新要使關鍵字的分類做得更好；但語者分類的 loss，卻加上了負號，也就是使參數更新反而要使語者的分類做得不好，其中 λ 為一個正數，必須由我們事前設定。這樣子的作法就是為了實現 feature disentanglement，藉由語者分類器的回饋，讓特徵抽取器不去學習抽取語者資訊的特徵，而學習只抽取文字資訊的

doi:10.6342/NTU202001860

特徵，使隱藏特徵能夠只有文字資訊，幫助關鍵字分類結果更好。除此之外，由於對抗的模型部分 (即語者分類器) 在最後進行關鍵字辨識時並不需要使用，因此模型的使用參數與純 TDNN 模型相同。

對抗式網路中分為三個部分，分別為 encoder、speech-classifier 與 speaker-classifier，我們希望透過使用不同的網路層來比較之間的差異，如 Figure 4.6 所示，語者分類器部分是固定不變的，特徵抽取器與關鍵字分類器則會替換 TDNN/CNN/LSTM 等，這樣總共會有 9 種模型會互相進行比較，並且我們也限制每個模型都使用相近的參數與訓練次數，以保證比較的公平性。

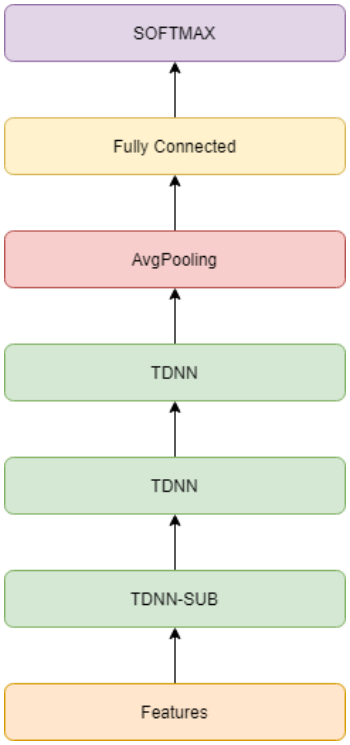


Figure 4.4: 端到端模型-純 TDNN 架構圖

4.2.3 訓練流程與參數設定

DNN-HMM 模型需要先訓練 GMM 模型，GMM 訓練的參數設定列於 Table 4.2，tri1 與 tri2 使用普通的 triphone 模型的訓練，tri3 加入線性判別分析 (linear discriminant analysis, LDA) 及最大似然線性轉換 (maximum likelihood linear transform, MLLT) 降維運算，tri4 與 tri5 加入了語者自適應訓練 (speaker adaptive training, SAT)。

doi:10.6342/NTU202001860

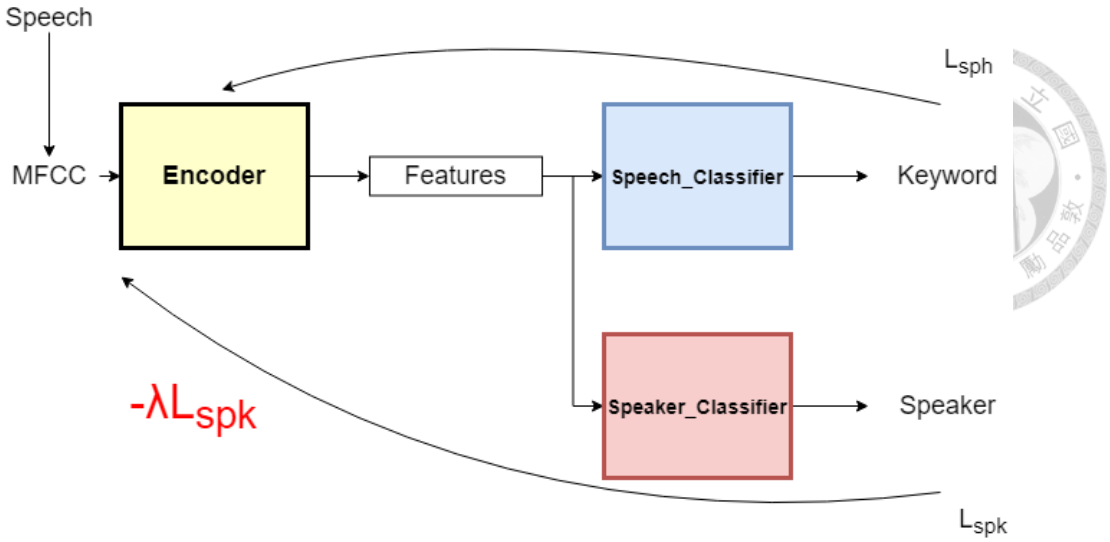


Figure 4.5: 端到端模型-對抗式訓練網路

model	script	num_leaves	tot_gauss	num_iters
mono	train_mono.sh	-	-	40
tri1	train_deltas.sh	250	350	35
tri2	train_deltas.sh	350	550	35
tri3	train_lda_mllt.sh	350	550	35
tri4	train_sat.sh	350	550	35
tri5	train_sat.sh	500	800	35

Table 4.2: GMM 訓練參數

訓練好第五個 triphone 模型 (tri5) 後，以此模型對訓練資料做強制對齊，訓練 DNN 模型，參數如下:

1. mini-batch size：128, 64
2. initial effective learning rate：0.001
3. final effective learning rate：0.001
4. epoch：2
5. sample-per-iter：1500000

端到端模型則直接使用抽出的 MFCC 特徵進行訓練，訓練的參數設定如下:

1. epoch：300

doi:10.6342/NTU202001860

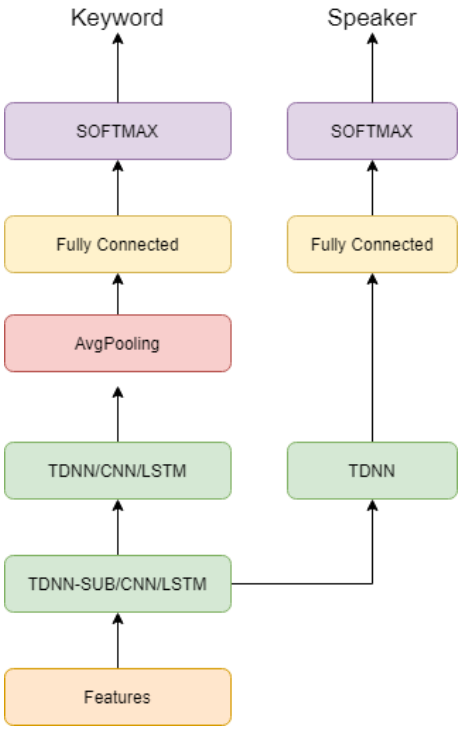


Figure 4.6: 端到端模型-對抗式訓練網路架構圖

2. batch size：32
3. initialization：Xavier initialization [23]
4. optimizer：Adam
5. learning rate：0.001
6. learning rate schedular：每 100 個 epoch 變為十分之一
7. loss：cross entropy

對抗式網路有兩個 loss，根據 DAT 介紹的方法，我們可以將 loss 表示為:

$$\theta_f \leftarrow \theta_f - \mu(\frac{\partial \mathcal{L}_y^i}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}_d^i}{\partial \theta_f}) \tag{4.1}$$

$$\theta_y \leftarrow \theta_y - \mu \frac{\partial \mathcal{L}_y^i}{\partial \theta_y} \tag{4.2}$$

$$\theta_d \leftarrow \theta_d - \mu\lambda \frac{\partial \mathcal{L}_d^i}{\partial \theta_d} \tag{4.3}$$

doi:10.6342/NTU202001860

其中 μ 是學習率 (learning rate)，λ 是超參數 (hyperparameter)，兩個分類器反覆地訓練。

4.2.4 效果評估方式

如此篇論文開頭所述，我們希望從多個方面進行效果評估，希望能更全面性的分析各個模型的優劣，一共有 5 種衡量標準:

1. 辨識錯誤率：即所有音檔辨識錯誤的比率，也是模型的訓練目標。
2. 參數數量：模型裡可變動 (訓練) 的參數數量。Table 4.3 列出我們的模型參數計算方式。
3. 乘法次數：神經網路內一連串的矩陣運算所進行的乘法次數。Table 4.3 列出我們的模型乘法次數計算方式。
4. 邊緣設備運算時間：將模型放置於邊緣設備，計算模型運算所需的實際時間。
5. DET (detection error tradeoff) curve：透過設定不同的閾值，計算 false alarm 與 false reject 作圖，曲線下面積越小越好。
6. 視覺化結果：將模型任一層的輸出利用 TSNE 投影，希望能看出去除語者資訊的效果。

我們以 TDNN 為例解釋參數數量與乘法次數的計算方式如 Figure 4.7，則我們可以寫出兩個式子為:

$$\#para. = W \cdot D_{in} \cdot D_{out} \tag{4.4}$$

$$\#mult. = W \cdot D_{in} \cdot D_{out} \cdot T_{out} \tag{4.5}$$

其中 W 為 kernel size、$D_{in}D_{out}$ 各為輸入輸出特徵的維度、T_{out} 是輸出的特徵數量。

doi:10.6342/NTU202001860

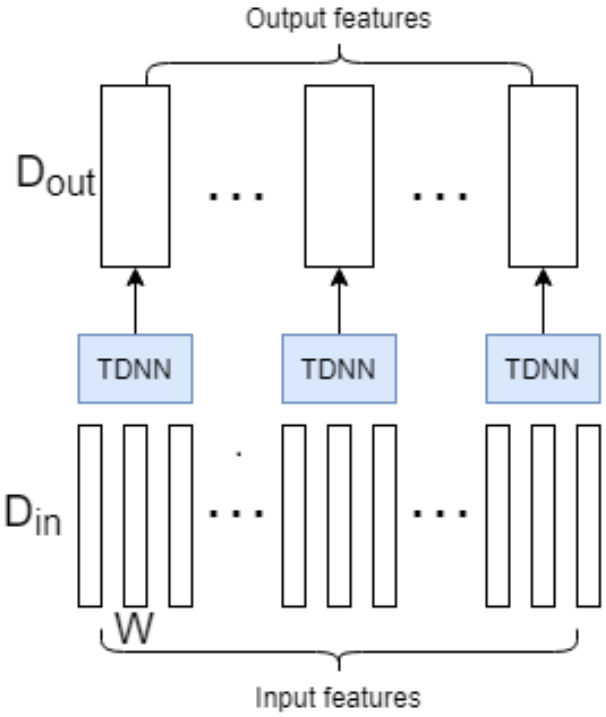


Figure 4.7: TDNN 參數計算圖

4.2.5 邊緣設備

將 DNN-HMM 與端到端模型訓練完畢後，我們將兩種模型分別透過 Vosk-api 與 Pytorch Mobile 安裝至手機，測試實際運作的時間，使用的手機是 Asus Zenfone Max Pro。

使用 Vosk-api 時，decoding 與平時不同，我們需要生成 HCL 與 G 兩個 graph，而不是 HCLG [24]，兩個 graph 在手機上 decoding 時會再合成為 HCLG，這麼做的原因主要有兩個:

1. HCL 與 G 分開時，兩者都不會太大，但合成為 HCLG 後卻是兩者相加後的好幾倍，邊緣設備上可能無法執行那麼大的 graph 或者嚴重降低效能。

2. 因為 HCL 與 G 分開，替換語言模型就變得非常有彈性。

核心程式部分是使用 C++ 進行實作。

Pytorch Mobile 的使用方法如 Figure 4.8 所示。我們需要將存起來的模型架構與參數直接儲存成 TorchScript module，不需要進行任何壓縮。核心程式部分是使用 JAVA 進行實作，只需要將輸入轉換成 JAVA tensor 即可。

doi:10.6342/NTU202001860

Layer	w	k	d	l	#Para.	#Mult.
Input	-	-	40	126	-	-
TDNN-SUB	3	3	32	42	3840	161280
TDNN	3	1	32	40	3072	122880
TDNN	3	1	32	38	3072	116736
Global Pooling	-	-	32	-	-	-
Linear	-	-	11	-	352	352
Softmax	-	-	-	-	-	-
Total	-	-	-	-	10336	401248

w 及 k 分別為 TDNN 的 kernel size 與 step size。d 及 l 分別為該層輸出特徵的長度及維度。#Para. 及 #Mult. 分別為該層所用到的參數量及矩陣的乘法次數。

Table 4.3: TDNN-AT 模型參數量計算

	Error rate					
Model	MFCC			Filter bank		
	train	validation	test	train	validation	test
單純 TDNN	1.5%	4.9%	4.6%	4.3%	7.5%	7.5%
對抗式訓練	1.9%	4.5%	4.3%	4.3%	8.2%	8.0%

Table 4.4: 對抗式訓練之進步及不同特徵的效果

4.3 實驗結果

4.3.1 對抗式訓練的效果

首先我們觀察對抗式訓練所帶來的進步，列於 Table 4.4，可以看出，透過對抗式訓練 validation 上進步了 4%，test 時進步了 3%。除此之外，我們也列出使用不同特徵的結果，MFCC 計算複雜，在特徵抽取時有較大的延遲，但獲得較好的結果；filter bank 雖然能較快的計算出來，但是結果並不好，因此我們最後選擇使用 MFCC 當作輸入特徵。訓練過程圖如 Figure 4.9 所示，對抗式訓練由於兩個 loss 交互的影響，導致一開始非常震盪，但後面得到了比較好的結果。訓練錯誤率來看，對抗式訓練也比較沒有那麼嚴重的 overfitting。

接著我們用視覺化來比較這兩個模型，我們透過 TSNE [25] 將 encoder (第一層)

doi:10.6342/NTU202001860

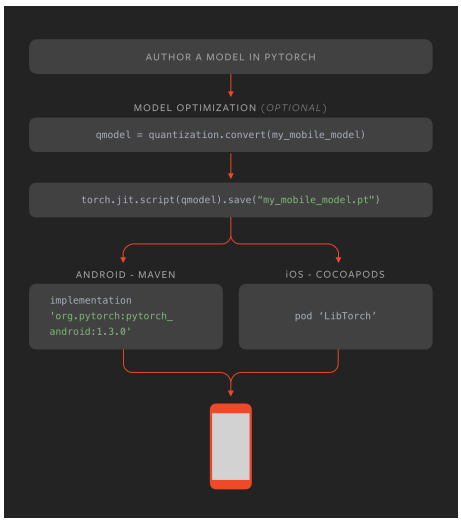
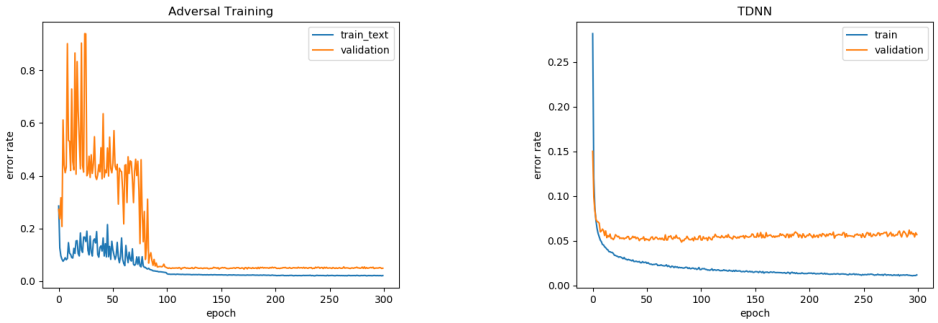


Figure 4.8: Pytorch Mobile 使用方法

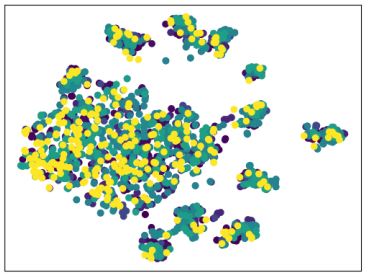
的 output 投影到二維平面，觀察它們的分布，如 Figure 4.10 所示 (圖 a、b)，圖中不同顏色的點代表不同語者。我們希望 encoder 能做到的是去除語者資訊，因此從兩張圖得比較可能可以看出，去除語者資訊後所有的語者在二維上會混在一起 (圖 b)，而原本的則會有一些分布的樣子 (圖 a)，這可能是一個能說明 feature disentanglement 的視覺化方法。此外我們也將最終層的結果畫出，如 Figure 4.10 所式 (圖 c、d)，圖中不同顏色的點代表不同 label，因此總共有 11 類，我們可以看出對抗式訓練的結果，各類比較集中，可能是導致結果較佳的原因，而兩圖中都有一小塊各種顏色混在一起，可能是品質不好的 data，導致模型都無法正確分類出來，因而混在一起。



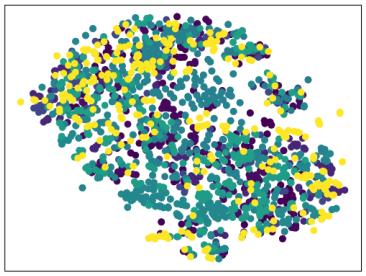
(a) 純 TDNN　　(b) 對抗式訓練

Figure 4.9: 訓練過程變化圖

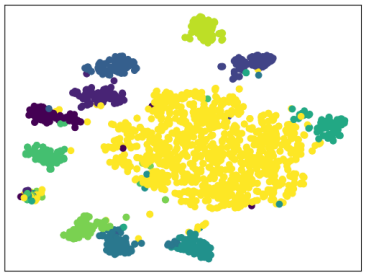
doi:10.6342/NTU202001860



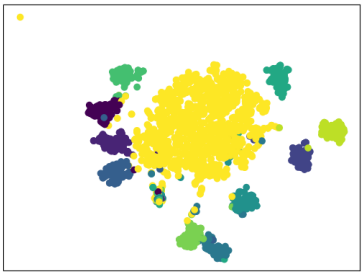
(a) 純 TDNN-第一層



(b) 對抗式訓練-第一層



(c) 純 TDNN-最終層



(d) 對抗式訓練-最終層

Figure 4.10: TSNE

4.3.2 不同網路層的效果

再來我們透過替換不同的網路層，且在所有的實驗設定皆相同下，比較之間的差異，列於 Table 4.5。TDNN 搭配不同網路層都獲得不錯的效果；LSTM 的訓練較慢，在 300 個 epoch 內可能因此無法做到太好；CNN 比起 TDNN 可以多考慮在頻率上的參數，但因為限制參數量得關係，並不能用太複雜的層，可能因此效果比較差一點。

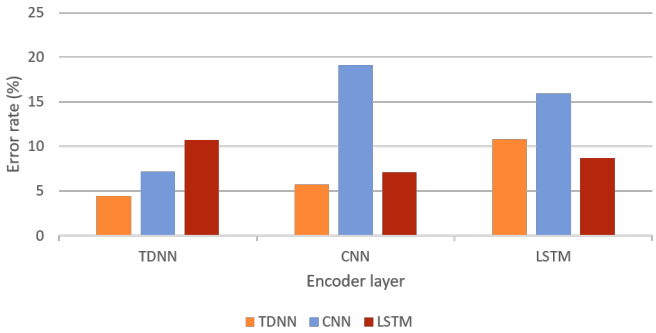


Figure 4.11: 不同網路層比較

doi:10.6342/NTU202001860

Encoder	Speech-Classifier	Error rate		#Para
		validation	test	
TDNN	TDNN	4.7%	4.4%	10K
TDNN	CNN	7.3%	7.2%	
TDNN	LSTM	9.9%	10.7%	
CNN	TDNN	5.2%	5.7%	
CNN	CNN	18.8%	19.1%	
CNN	LSTM	6.9%	7.1%	
LSTM	TDNN	11.1%	10.8%	
LSTM	CNN	15.4%	15.9%	
LSTM	LSTM	8.9%	8.7%	

Table 4.5: 不同網路層比較

	Error rate				#para
Model	Use i-vector		No i-vector		
	validation	test	validation	test	
單純 TDNN	11.0%	12.1%	4.9%	4.6%	10K
對抗式訓練	13.1%	14.1%	4.5%	4.3%	10K
DNN-HMM	2.7%	2.1%	2.6%	1.9%	7.8M
DNN-HMM	7.6%	6.9%	9.2%	7.9%	10K

Table 4.6: DNN-HMM 與端到端模型的比較

4.3.3 DNN-HMM 與端到端的比較

這裡討論 DNN-HMM 與端到端模型的差異，列於 Table 4.6，其中我們也比較是否使用 i-vector 的影響，DNN-HMM 方法常會使用 i-vector 來提供語者資訊，端到端部分我們有用對抗式去掉語者資訊，但為了比較公平，我們將兩個模型與兩種特徵都實作來比較。可以看出 DNN-HMM 雖然能作到很好，但需要使用大量的參數，但如果我們將參數限定至與端到端相同時，DNN-HMM 的表現較差。使用 i-vector 時，端到端的模型皆變得不好，我們認為是因為模型參數太少，無法擬何維度較大的特徵。DNN-HMM 部分，參數多時 i-vector 似乎沒什麼幫助，參數少時有一點進步，可能是因為對於 DNN-HMM 本身模型實在太小，i-vector 能多提供一點資訊，幫助辨識。

doi:10.6342/NTU202001860

Model	Error rate		#Para	#Mult
	validation	test		
Resnets15 [2]	-	4.2%	238K	894M
TDNN-SWSA [3]	-	4.19%	12K	402592
TDNN-SWSA*	5.5%	5.6%	12K	402592
單純 TDNN	4.9%	4.6%	10K	401248
對抗式訓練	4.5%	4.3%	10K	401248
DNN-HMM	2.7%	2.1%	7.8M	387M
DNN-HMM	7.6%	6.9%	10K	1.6M

* Reimplementation

Table 4.7: 與其他論文比較

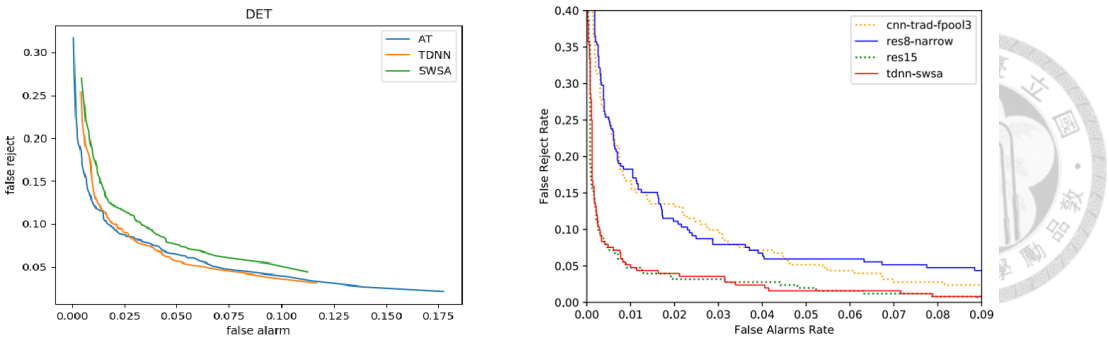
4.3.4 與其他論文比較

最後就各方面與其他論文結果比較，列於 Table 4.7。與 ResNets15 相比，我們的模型雖然在錯誤率上多了 0.1%，但參數與乘法次數遠遠的少於它。與 TDNN-SWSA 相比，我們提出的模型雖然在錯誤率上多了 0.11%，但參數量少了 2000，或許看起來不多，但在實際運算時間上，仍有不小的差異。除此之外，我們也用 DET curve 來比較各個模型的表現如 Figure 4.12，左邊為我們的模型、右邊為其他的模型，因為無法重現結果所以只能借用其他論文上的圖。在我們的模型中可以看出，對抗式訓練的 DET 曲線下的面積比較小，也就是比較好的模型，但在閾值比較小的時候，純 TDNN 有比較低的 FAR，我們認為這是因為純 TDNN 對於 filler 的分類較佳，因此就算在低閾值時，filler 的信心指數仍然很高，因此也不會誤喚醒，但整體來說，對抗式的結果稍微好一點。那與其他模型比較時，我們可以以 false reject rate 在 0.05 時為標準，我們的模型得到 0.05 的 false alarm rate，其他模型則是 0.01，這部分是我們模型較弱的地方。

4.3.5 邊緣設備運行時間

邊緣設備之間的比較列於 Table 4.8，運算時間是 100 次實驗結果平均所得。首先我們觀察模型計算時間，對抗式訓練比 TDNN-SWSA 時間短了 0.9ms，應該就是參數量所造成的影響。Resnets 的結果來自其他論文，並且是使用 Tensorflow

doi:10.6342/NTU202001860



(a) our models (b) other models

Figure 4.12: DET curve [3]

Model	特徵抽取 (1 秒音訊)	模型計算	Total time
Resnets15 [26]	-	0.8ms	-
TDNN-SWSA*	118ms	2.0ms	120ms
對抗式訓練 *	118ms	1.1ms	119.1ms
DNN-HMM**	-	-	120ms

* 利用 Java 實作。
** 利用 C++ 實作。

Table 4.8: 邊緣設備運算時間比較

Lite 實作，可以看到時間少了 0.3ms，這個部分我們認為是 Pytorch Mobile 與 Tensorflow Lite 之間的差異。接著我們觀察特徵抽取時間需要 118ms，幾乎就是總時間，這個有幾個可能的原因:

1. JAVA 與 C++ 實作的差異。

2. 沒有使用 Online 的特徵抽取。

所謂的 Online 就是錄音時就不斷進行特徵抽取，而不是錄完 1 秒音檔後才做，兩者會有一定的時間差異。最後綜觀比較，端到端模型如果能克服特徵抽取時間，在時間與空間上比起 DNN-HMM 都是較為優勢的。

4.4 錯誤分析與討論

前面我們透過訓練過程圖、TSNE 等想來解釋對抗式訓練帶來的效果，尤其是 TSNE 投影到二維的視覺化技巧，可能可以說明去除語者資訊後特徵的樣子，但

doi:10.6342/NTU202001860

這個證明有可能不夠強力，有可能無法完全說明其帶來的效果，或許有更好的方法可以說明結果變好的原因。

DNN-HMM 與端到端模型的結果之間，其實仍有非常大的差異，因此我們認為端到端模型仍舊沒有做到最好，應該可以再使用差不多的參數下，做到更低的錯誤率。而對於 DNN-HMM 來說，應該也可以再將參數量減少，犧牲一點錯誤率，使用到的參數能夠降低。

對抗式訓練模型與 TDNN-SWSA 之間的比較，參數量雖少了 2000，但乘法次數其實是差不多的，因此可能要用實際運行時間來比較。最終我們也在手機裝置上得到模型計算時間少了 0.9ms，確實參數量的減少是有幫助的。而端到端模型與 DNN-HMM 有著差不多的總時間，這一點可能是最需要改善的，端到端的特徵抽取時間成為了瓶頸，模型計算時間端到端模型取得了非常大的優勢，因此勢必要改善特徵抽取時間。

我們透過 confusion matrix 來錯誤分析如 Figure 4.13 所示，縱軸是正確的 label、橫軸是模型預測的 label，數字代表某個關鍵字辨識成某個關鍵字的機率，因此我們可以從圖中觀察哪些 label 間容易互相混淆。端到端模型 (圖 a、b) 的錯誤有兩個傾向:

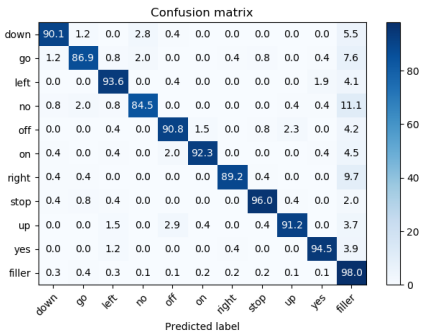
1. 錯誤時大多辨識成 filler。

2. 單音節的字辨識率較低，也較容易互相混淆。

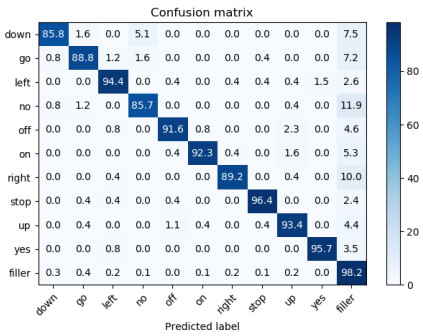
第一點是由於 filler 佔訓練資料較多所導致，解決方法是設定 loss 的權重，使每個關鍵字所佔的重要程度相同，但經過我們的實驗，結果對整體的辨識率並沒有太大的影響 (圖 d)。第二點可能是單音節的字念的時候並沒有什麼變化，加上單音節的字之間相似度也較高 (像是 go 與 no)，單音節的字之間的混淆程度都比較高，在 DNN-HMM (圖 c) 中似乎也有一點這種傾向但較不明顯。此外，DNN-HMM 各個 label 的辨識率也較平衡，沒有哪一類辨識率特別高。

最後我們想討論訓練與測試的差異如 Figure 4.13 所示 (圖 e、f) ，在訓練時每個關鍵字結果其實都是差不多的，但我們可以看到有一些關鍵字在測試時變得非常差，那些字也都幾乎是前面所提到的單音節的字，因此對於這些字我們可能需要做一些特殊的方法來改進。

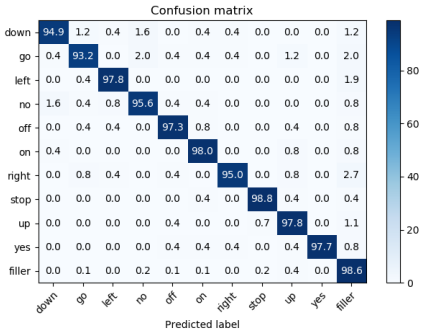
doi:10.6342/NTU202001860



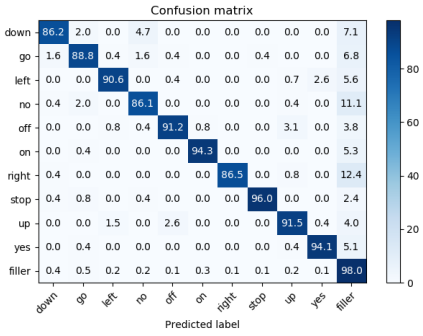
(a) 純 TDNN



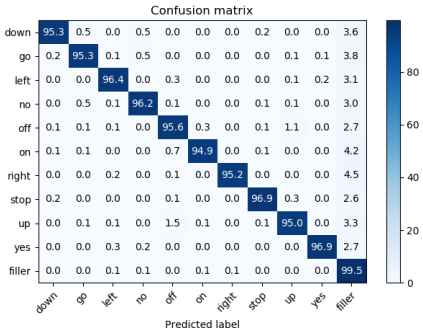
(b) 對抗式訓練



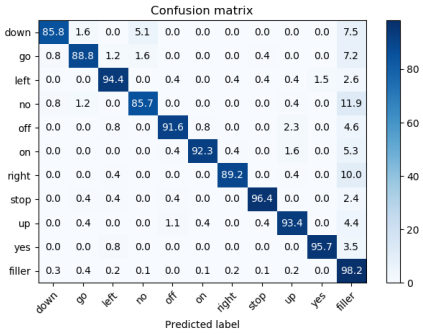
(c) DNN-HMM



(d) 對抗式訓練-filler 權重調整



(e) 訓練



(f) 測試

Figure 4.13: Confusion matrix

doi:10.6342/NTU202001860

Chapter 5

結論與未來展望

5.1 結論

以下為本次基於邊緣計算之語音喚醒系統研究中，我們對各項實驗結果進行分析後，歸類出的結論與我們所做的改進與貢獻:

1. 在大部分的語音喚醒論文中，往往只以參數數量與乘法次數作為評斷是否符合 small-footprint 的需求，較少論文透過將模型安裝至邊緣設備直接測試實際的延遲時間。我們除了分析模型的架構、計算乘法次數，也實際將模型安裝至邊緣設備，做更全面的比較。並且也比較了 DNN-HMM 與端到端模型在各個方面的差異，也是較少論文有研究的，因此能夠觀察到所有不同的模型與不同的實作方式所帶來的優劣。

2. 利用對抗式訓練獲得 feature disentanglement 的效果，使模型獲得更低的錯誤率，並且對抗式的部分在測試時並不需要使用，因此能夠擁有與原本模型相同的參數，不會造成負擔，能夠符合 small-footprint 的需求，同時改善模型的表現，一舉兩得。

3. 在相同的訓練設定下，我們比較了常見的網路層 TDNN、CNN、LSTM 的表現，TDNN 算是得到最好的表現，我們認為在語音喚醒中喚醒詞都是較簡單的字，TDNN 考慮時間的變化已經足夠，CNN 多考慮了頻率軸的變化、LSTM 可能需要更長序列來獲得更好的結果。

doi:10.6342/NTU202001860

4. 我們的模型在錯誤率上達到 4.3%，與 ResNets15 (4.2%) 、TDNN-SWSA (4.19%) 差距不大，參數數量與乘法次數也都小於上述兩個模型。除此之外，在邊緣設備上的運算時間，我們的模型也達到 1.1ms，比 TDNN-SWSA (2.0ms) 少了將近一半。而 DNN-HMM 能有差不多總時間的原因可能是 C++ 實作，尤其特徵抽取時需要複雜的計算，可能導致 JAVA 與 C++ 產生很大的差距。

5. 邊緣設備上的運行時間受很多方面影響，特徵抽取是否是 online 抽取很大程度影響運行時間；不同的套件優化的程度也可能不同 (Tensorflow Lite 與 Pytorch Mobile) ；DNN-HMM 模型因為需要 decoding，因此控制 decoding 使用的 graph 大小，則是影響運行時間的關鍵。

6. 我們透過 confusion matrix 觀察錯誤率高的字是哪些，端到端模型幾乎都有單音節字較差的傾向，這可能是要再多加研究改善的部分。filler 的辨識率都比較高，調整權重也無法產生太大的影響，但換個角度思考，將其他字訓練得更好提高整體辨識率可能才是解決之道。

5.2 未來展望

1. 端到端實作部分目前使用 JAVA 導致特徵抽取時間過長，未來希望能用 C++ 或者更底層的語言實作，減少特徵抽取所花的時間。

2. 我們觀察到端到端的錯誤結果有一些傾向，但我們並不知道原因為何。端到端的模型缺點之一就是較難分析裡面的錯誤，現在有一些方法像是 attention 可以顯示模型學到的 align 結果，或許是一個方式可以幫助我們分析錯誤，因此未來希望透過此方式將結果做得更好。

3. 目前語音喚醒通常都是使用事先定義好的關鍵字來訓練，如果要使其更有彈性，可能需要能夠讓使用者定義 keyword，但如此勢必要重新訓練，且訓練資料是非常少的，因此需要使用到 few-shot learning 的技術 [27] [28]，這個是未來可以發展的目標。還有一個方法是用 sequence to sequence 模型來解決，也就是語音辨識模型，這類模型雖然很大，但壓縮技術的進步及端到端

doi:10.6342/NTU202001860

模型仍然比 DNN-HMM 模型來的快，在邊緣裝置上速度也不會太差，像是 Google 的語音辨識就是用此類模型。

4. 我們將模型安裝到手機裝置測試實際時間，未來我們也希望可以安裝至嵌入式系統，也想嘗試使用整數運算，雖然這樣絕對會導致錯誤率上升，但也可以使延遲時間下降，兩者之間的權衡也是一大學問。

doi:10.6342/NTU202001860

Bibliography

[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4087–4091, IEEE, 2014.

[2] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5484–5488, IEEE, 2018.

[3] Y. Bai, J. Yi, J. Tao, Z. Wen, Z. Tian, C. Zhao, and C. Fan, "A time delay neural network with shared weight self-attention for small-footprint keyword spotting," *Proc. Interspeech 2019*, pp. 2190–2194, 2019.

[4] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[5] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[6] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

doi:10.6342/NTU202001860

[7] D. R. Miller, M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, and H. Gish, "Rapid and accurate spoken term detection," in *Eighth Annual Conference of the international speech communication association*, 2007.

[8] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 615–622, 2007.

[9] S. Parlak and M. Saraclar, "Spoken term detection for turkish broadcast news," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5244–5247, IEEE, 2008.

[10] R. C. Rose and D. B. Paul, "A hidden markov model based keyword recognition system," in *International conference on acoustics, speech, and signal processing*, pp. 129–132, IEEE, 1990.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[12] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, IEEE Signal Processing Society, 2011.

[13] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[15] D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, "A time-restricted self-attention layer for asr," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5874–5878, IEEE, 2018.

doi:10.6342/NTU202001860

[16] M. Sperber, J. Niehues, G. Neubig, S. Stüker, and A. Waibel, "Self-attentional acoustic models," *arXiv preprint arXiv:1803.09519*, 2018.

[17] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[18] O. Irsoy and C. Cardie, "Deep recursive neural networks for compositionality in language," in *Advances in neural information processing systems*, pp. 2096–2104, 2014.

[19] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," 2014.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[21] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[22] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks.," in *Interspeech*, vol. 2013, pp. 2345–2349, 2013.

[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[24] C. Allauzen, M. Riley, and J. Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," 2009.

[25] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

doi:10.6342/NTU202001860

[26] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal convolution for real-time keyword spotting on mobile devices," *arXiv preprint arXiv:1904.03814*, 2019.

[27] Y. Chen, T. Ko, L. Shang, X. Chen, X. Jiang, and Q. Li, "An investigation of few-shot learning in spoken term classification," *arXiv*, pp. arXiv–1812, 2018.

[28] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, pp. 4077–4087, 2017.

doi:10.6342/NTU202001860