Introduction to Artificial Intelligence Final Project: Voice Recognition

PowerPoint Presentation

Introduction

Our Problem

Our project aims to train a robust acoustic fingerprint recognition model using only a small amount of data. Typically, acoustic fingerprint recognition models require a large amount of data to achieve good performance. However, we leverage the capability of the voice generation model called Bark, which can generate corresponding sounds based on given voices. By using the sounds generated by Bark as our data source, we can achieve the goal of training a high-quality acoustic fingerprint recognition model with minimal data. We then apply this model to real-life voice recognition tasks.

Our acoustic fingerprint recognition model consists of two parts: a small model for determining acoustic similarity and a larger model that serves as a wrapper for classification decisions.

Most of our experiments focus on exploring methods to improve the accuracy of the small model in determining acoustic similarity. To enhance the accuracy of the small model, we conducted various experiments. For example, we compared the use of LSTM and GRU in the RNN part and examined the impact of setting the number of RNN layers to 1 or 2. We also explored different approaches such as using Conv1d with the original audio sequences and Conv2d with Mel Spectrograms to test which methods could achieve higher accuracy with limited data. Finally, we leverage a linear classifier model composed of DNN, ReLU and Dropout layers to output a value that can be understood as the similarity between two given voices. As for the larger model that serves as a wrapper for classification decisions of the small model, we implemented it using Softmax.

我們的專案是要利用少量的資料就能夠訓練出良好的聲紋辨識模型。一般的聲紋辨識模型需要利用大量的資料才能夠訓練出良好的聲紋辨識模型，但是我們透過人聲生成模型 Bark 能夠使用給定的聲音生成出對應的聲音的這個特性，利用 Bark 所生成的聲音當作資料來源，達成只需要少量的資料就能夠訓練出良好的聲紋辨識模型的目標，並且將這個模型套用到真人的聲音辨識上。我們的聲紋辨識模型總共分成兩個部分，分別是判定聲音相似度的小模型以及作為包裝器（wrapper）進行分類判定的大模型。

我們所做的實驗大多是在嘗試有什麼方法可以讓判定聲音相似度的小模型能夠達成更好的正確率。為了讓小模型能夠有更高的準確率，我們做了許多實驗，像是在 RNN 的部分比較了使用 LSTM 和 GRU，以及比較 RNN 的層數設置為 1 和 2 對於結果的差別，還有使用原始聲音序列的 Conv1d 和使用 Mel Spectrogram 的 Conv2d 等多種方法，來測試哪些方法可以達成在只有少量資料的情況下，就能夠訓練出高準確度的模型。最後再透過由 DNN、ReLU 和 Dropout 組成的線性模型，輸出一個可以被近似理解為聲音相似度的數值。至於作為小模型的包裝器（wrapper）進行分類判定的大模型，我們使用 Softmax 來進行實作。

Related work

- **Speech recognition with deep recurrent neural networks**
- **Convolutional Neural Networks for Speech Recognition**
- **Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions**

- 第一篇在講 RNN 在聲音辨識領域的應用，我們參考了他們 LSTM 和 雙向 RNN 的想法。
- 第二篇在講 CNN 在語音辨識領域的應用，我們參考了他們使用頻譜圖和 CNN 架構的想法。
- 第三篇在講使用 Mel Spectrogram 的語音合成，我們參考了他們使用 Mel Spectrogram 的想法。

Dataset/Platform

The training data we used was obtained through a voice generation model called Bark. Bark is a model that takes text input and provides corresponding voice samples as output. In our implementation, we first used Bark to generate five distinct male voices and five distinct female voices by inputting a text paragraph. We then further generated more voice segments using these ten different voices. However, due to hardware limitations, we were only able to generate 30 segments for each voice in our implementation.

For testing the voice recognition with real human voices, we sourced the data from a Talk at Google interview video on YouTube titled "Guardians of the Galaxy." We collected voice segments from Chris Pratt and Vin Diesel. These voice clips were gathered to conduct real-world tests and evaluate whether our model can accurately distinguish between voices generated by Bark and voices of real individuals.

我們使用的訓練資料是透過人聲生成模型——Bark 獲得的。Bark 是一個能夠輸入文字段落和提供的聲音來生成聲音片段的模型。在我們的實作中，我們先透過輸入一串文字讓 Bark 生成五位不同的男性聲音以及五位不同的女性聲音，並透過這十位不同的聲音再進一步生成出更多的聲音片段。但是由於我們的硬體限制，我們的實作中只有將每個人聲各生成出 30 個片段。

至於我們用於測試真人的聲音辨識的資料來源是從 Talks at Google 的 Youtube 中的一部訪談影片 Guardians of the Galaxy，蒐集了Chris Pratt 以及 Vin Diesel 的聲音片段。收集這些聲音片段是為了後續的真人測試上，用來測試我們的模型是不是能夠準確判斷不是由Bark所生出來的聲音，而是真人的聲音。

Baseline

VoiceDataset

VoiceDataset is the preprocessing part and sampling in our implementation. It inherited the Dataset class from torch.utils.data.

For the preprocessing part, it will read voice file from the disk under specific directories, and maybe do some

```
class VoiceDataset(Dataset):
  def __init__(self, directory, embedding_dim = EMBEDDING_DIM):
    super().__init__()
    self.k = 1
    self.population = 0
    self.length = 0
```

preprocessing depends on whether Conv1d or Conv2d is used.

As for the sampling part, it will randomly pick a target, and generates one-hot encoding labels corresponding to the chosen voice. Then it will iterate all voices in the dataset to return sample-target pairs to use later but were stored in the different list now.

VoiceDataset 是我們實作中用來對資料進行預先處理以及採樣的一個Class。它繼承了 torch.utils.data 中的 Dataset Class。

在資料的預先處理部分，它會從特定目錄下讀取聲音文件，並根據是否使用 Conv1d 或 Conv2d 進行一些預先處理的操作。

至於採樣部分，它先會隨機選擇一個目標，然後生成對應所選聲音的 one-hot encoding 的標籤。接著他會跑過資料集中的所有聲音，再 return 回樣本以及目標的 pair，讓接下來的模型可以使用，不過它們會先被分別儲存在不同的列表中。

```
    self.embedding_dim = embedding_dim
    self.voices = []
    self.base_map = {}
    for filename in os.listdir(directory):
      if not filename.endswith(".wav"): continue
      self.voices.append([])
      self.population += 1
      folder_name = filename.split(".wav")[0]
      self.base_map[folder_name] = self.population
      for fn in os.listdir(f"./data/bark/{folder_name}"):
        if not fn.endswith(".wav"): continue
        self.voices[-1].append(load_voice(f"./data/bark/{folder_
        self.length += 1

  def __getitem__(self, _):
    samples, targets, labels = [], [], torch.zeros(len(self.voic
    r_index = random.randint(0, len(self.voices) - 1)
    labels[r_index] = 1
    for person in self.voices:
      for s in random.choices(person, k=self.k):
        r = random.randint(0, len(s) - self.embedding_dim)
        samples.append(s[r:r + self.embedding_dim])
      for t in random.choices(self.voices[r_index], k=self.k):
        r = random.randint(0, len(t) - self.embedding_dim)
        targets.append(t[r:r + self.embedding_dim])
    samples = torch.tensor(np.concatenate(samples)).view(len(sel
    targets = torch.tensor(np.concatenate(targets)).view(len(sel
    return samples, targets, labels

  def get_selected(self, target):
    samples, targets = [], []
    for person in self.voices:
      for s in random.choices(person, k=self.k):
        r = random.randint(0, len(s) - self.embedding_dim)
        samples.append(s[r:r + self.embedding_dim])
        r = random.randint(0, len(target) - self.embedding_dim)
        targets.append(target[r:r + self.embedding_dim])
    samples = torch.tensor(np.concatenate(samples)).view(len(sel
    targets = torch.tensor(np.concatenate(targets)).view(len(sel
    return samples, targets

  def __len__(self): return self.length
```

Judge

Judge is the main part for training in the architecture of our model design. It consists of two Sequential layers used for passing features in the voices and a LSTM used for RNN. Certain ratio is assigned for Dropout to prevent overfitting. In the end of seq2 , we apply a Sigmoid function to obtain a float between 0 and 1, which is similar to the concept of similarity. When doing forward the output is collected for concatenation and return as a tensor of 1D shape 10.

This is a fully connected CNN/DNN sub-model, and is not customizable.

The reason why we use window size 25ms and step stride 10ms is according to this video.

Judge 是我們這份 Final Project 最主要訓練的部分。它是由兩個 Sequential 層組成，用來傳遞聲音中的特徵，以及一個用於 RNN 的 LSTM 層。我們分配了一定比例的 Dropout 來防止 Overfitting。在 seq2 的最後，我們利用 Sigmoid 函數來獲得一個介於 0 和 1 之間的 float number，這就像是相似度的概念。在 forward 的時候，輸出會被收集起來進行連接，並返回形狀為 1D 的 tensor。

這是一個無法自定義的完全連接 CNN/DNN 子模型。

至於我們將 window size 設置成 25 毫秒 以及 step stride 設置成 10 毫秒是根據李宏毅教授在一部影片所提到的，我們有將連結放在 PPT 中。

```
class Judge(nn.Module):
  def __init__(self, k):
    super().__init__()
    self.k = k
    self.seq1 = nn.Sequential(
      nn.Conv1d(1, 64, int(0.025 * SAMPLE_RATE), int(0.01 * S
AMPLE_RATE)),
      nn.BatchNorm1d(64),
      nn.MaxPool1d(4),
      nn.ReLU(True),
      nn.Conv1d(64, 128, 4),
      nn.BatchNorm1d(128),
      nn.MaxPool1d(4),
      nn.ReLU(True),
      nn.Conv1d(128, 256, 4),
      nn.BatchNorm1d(256),
      nn.MaxPool1d(4),
      nn.ReLU(True),
      nn.Flatten(),
      nn.Dropout(0.3)
    )
    self.seq2 = nn.Sequential(
      nn.Linear(256 * 3 * 2, 1024),
      nn.ReLU(True),
      nn.Dropout(0.2),
      nn.Linear(1024, 256),
      nn.ReLU(True),
      nn.Dropout(0.1),
      nn.Linear(256, 64),
      nn.ReLU(True),
      nn.Linear(64, 1),
      nn.Sigmoid(),
    )

  def forward(self, sample, target):
    outputs1 = self.seq1(sample.view(-1, 1, sample.size(1)))
    outputs2 = self.seq1(target.view(-1, 1, target.size(1)))
```

```
    outputs = self.seq2(torch.cat((outputs1, outputs2), dim=
1))
    outputs = torch.cat([outputs[self.k * i:self.k * (i + 1),
:] for i in range(0, outputs.size(0), self.k)], dim=1).sum(0)
    return outputs
```

Model

Model is a class that is designed for wrapping Judge and Softmax layer for the purpose of handling varying batch size. In the Model part, the batch size is the actual batch size from the batched data given by the dataloader. However, in the Judge part, the batch size is the number of people, which enables the whole model structure to deal with multiple people so that we can avoid modifying the Judge model again and again with only the number of people changed. Instead, by leveraging this method but not hardcoded the number of people into the Judge, we can augment the model easily and maximize its power.

Model 是一個用於包裝 Judge 和 Softmax 層的 Class，用來處理不同的 batch size。在 Model 部分，batch size 是用 data loader 所提供的batched data 的實際大小。然而，在 Judge 部分，batch size 是人數的數量，讓整個模型結構能夠處理多人的情況，避免如果只改變人數就需要反覆修改 Judge 模型。通過利用這種方法，而不是將人數硬寫到 Judge 中，我們可以輕鬆擴充模型並最大程度地發揮他的功能。

```
class Model(nn.Module):
  def __init__(self, n, k = 1):
    super().__init__()
    self.n = n
    self.k = k
    self.judge = Judge(k)
    self.softmax = nn.Softmax(dim=1)

  def forward(self, samples, targets):
    outputs = []
    for s, t in zip(samples, targets): outputs.append(self.ju
dge(s, t).unsqueeze(1))
    outputs = torch.cat(tuple(outputs), dim=1).permute(1, 0)
    outputs = self.softmax(outputs)
    return outputs
```

Concatenating the parts mentioned above, we get a model that judges similarity between voices, hence later processes for training can be implemented.

將上述提到的兩個部分接在一起，我們就得到了一個用於判斷聲音相似度的模型，並且可以實作後續的訓練。

Main Approach

Our training approach involves using sounds generated by Bark for training purposes. In each training iteration, we randomly sample a specific duration of sound, with a chosen duration of 3 seconds as our input. The sampling process is implemented within the VoiceDataset class, which allows for modifications of the duration as needed.

我們的訓練方法是利用 Bark 生成的聲音來進行訓練。在每個訓練周期中，我們會隨機抽樣一段特定的聲音時長，而我們是選擇 3 秒作為輸入。至於抽樣過程是在 VoiceDataset 的 Class 中實現的，我們可以根據需求，對採樣的時長進行修改

Techniques

In our implementation, we utilize LSTM and GRU as the architectures for the RNN component. Their designs enable the model to capture long-term dependencies in sequential data. Next, we will introduce and briefly compare the differences between LSTM and GRU.

在我們的實作中，我們利用 LSTM 和 GRU 作為 RNN 的架構。它們的設計使得模型能夠捕捉到序列資料中的長期 dependency，接下來我們將會介紹並且簡單比較他們之間的差異。

LSTM

LSTM consists primarily of a memory cell and three types of gates: the forget gate, the input gate, and the output gate.

The memory cell acts as a pathway for information to flow through. During sequence processing, the memory cell retains relevant information and carries it forward, preserving earlier information.

As for the gates, the forget gate takes the previous hidden state information and the current input and passes them through a sigmoid function, using the output to determine

what to forget or remember from the previous information. The input gate also takes the previous hidden state information and the current input, passing them through a sigmoid function to determine their importance and update the cell state. The output gate determines the value of the next hidden state to be used in the future, incorporating relevant information from the previous inputs.

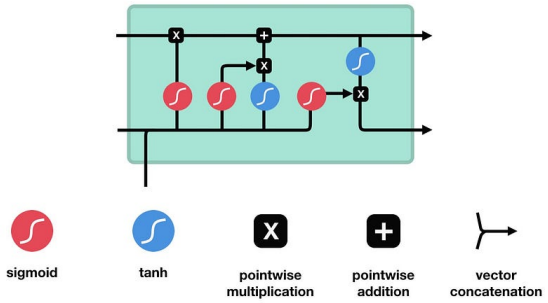
Due to the relatively complex structure of the memory cell and gates in LSTM, it requires more computation time and takes longer to converge during training.

LSTM 主要由記憶單元和三種類型的門組成：遺忘門、輸入門，以及輸出門。

記憶單元是一個讓訊息傳遞下去的通路。在序列處理過程中，記憶單元能一直攜帶著相關訊息，把較早的訊息保留下來。

至於門的部分，遺忘門會將先前隱藏狀態的資訊和當前輸入的信息同時輸入到 Sigmoid 函數，並用輸出值來決定是否要遺忘或記住先前的訊息。輸入門也會將先前隱藏狀態的訊息和當前輸入的訊息輸入到 Sigmoid 函數來決定重要性並用來更新記憶單元的狀態。輸出門則決定未來使用的下一個隱藏狀態的值，隱藏狀態中包含了先前輸入的相關信息。

由於 LSTM 具有相對複雜的記憶單元和門的結構，所以會需要花費更多時間計算，在訓練時也需要更長的時間才能收斂。

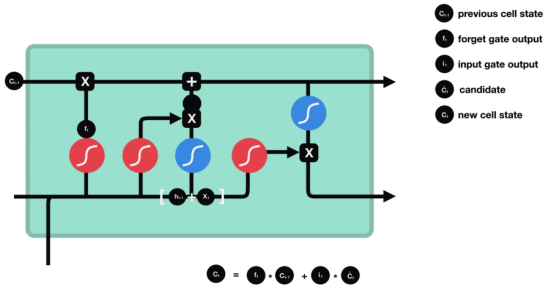


💡 **LSTM is used to capture the recurrent features presented in the data.**

GRU

Similar to LSTM, but with improved execution speed and reduced memory usage, GRU simplifies the structure of gates. While LSTM has three gates (forget gate, output gate, and input gate), GRU has two gates, namely the update gate and reset gate. The update gate in GRU is similar to the forget gate and input gate in LSTM as it determines what information to discard and what new information to add. The reset gate, on the other hand, determines how much previous information to discard. GRU combines the cell state and hidden state (h_t) and computes new information in a different way compared to LSTM.

GRU 和 LSTM 很像，但是 GRU 提高了執行速度並且減少記憶體的用量。他簡化了門的結構，將 LSTM 的三個門簡化成兩個門，分別是更新門 (reset gate) 與重置門 (update gate)。更新門類似 LSTM 的遺忘門和輸入門，它是用來決定丟棄什麼信息和添加什麼新信息。重置門的作用是用來決定要丟棄多少先前資訊。另外 GRU 把單元狀態 (cell state) 和隱藏狀態 (h_t) 進行合併，計算新資訊的方式也和 LSTM 也有所不同。



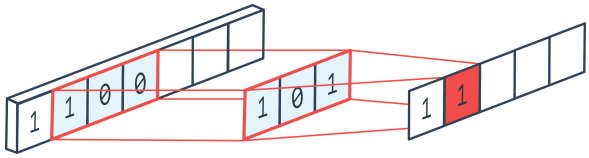
💡 **GRU is used to capture the recurrent features presented in the data.**

Conv1d

The original form of WAV file is a sequence of amplitude, where sequence indexes represent timestamps. Conv1d takes in 1-dimensional sequence of sound and extract the features during training; the input passes through kernel

layer and gives an output. In our implementation, the first input is the sequence of sound, where the sample rate is 24000 Hz with a duration of 3 seconds, therefore a size of 72000 in total. This sequence is fed into seq1 for later processing such as ReLU, MaxPool1d, etc.

WAV 檔案的原始形式是一個振幅序列，其中序列的 index 代表著時間戳記。Conv1d 會先接收一維聲音序列，並在訓練過程中提取特徵。輸入會通過 kernel 層並產生輸出。在我們的實作中，第一個輸入是聲音序列，採樣率是 24000 Hz，持續時間為 3 秒，因此總大小為 72000。這個序列會先被傳入 seq1 進行後續處理，如 ReLU、MaxPool1d 等。

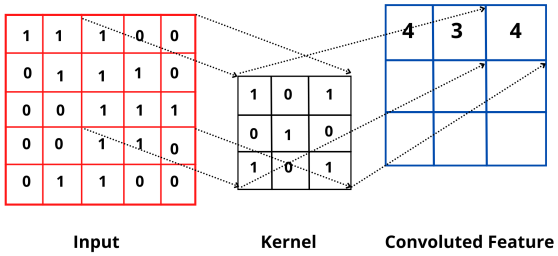


Conv1d is used to capture the sequence features presented in the data amplitude time series.

Conv2d

Since the feature is not obvious in wave form and the linear form of amplitude and frequencies need a logarithmic form for a better training performance, we adopt Mel Spectrogram, which is a 2-dimensional sequence and requires Conv2d to be fed into the model. Conv2d is similar to Conv1d, while the input, kernel, and output parts are all 2-dimensional. Conv2d is commonly used in image training, and the spectrogram we adopt is also of 2D type, therefore we leverage Conv2d for later processing.

由於聲音特徵在 Conv1d 的波型圖中不明顯，而且振幅和頻率需要由線性的關係轉換為對數形式，才能夠獲得更好的訓練效果以及貼合人類的實際感知聲音方式，因此我們採用 Mel Spectrogram 來轉換資料，並通過 Conv2d 輸入模型。Conv2d 與 Conv1d 類似，只是輸入、kernal 和輸出部分都是二維的。Conv2d 通常用於圖像訓練，而我們採用的頻譜圖也是二維數值資料，因此會使用 Conv2d 進行後續處理。

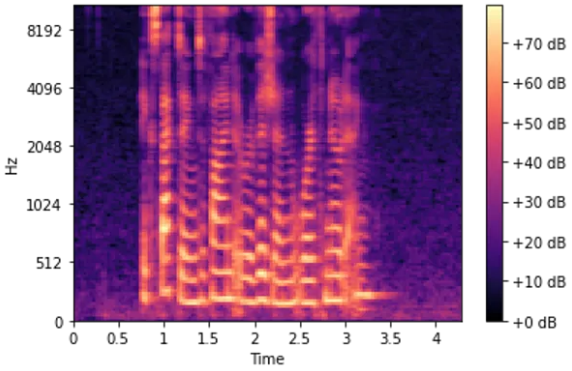


Conv2d is used to capture the area features presented in the converted data Mel Spectrogram.

Mel Spectrogram

Spectrogram is a conversion from voice to image. Using Fourier transform, it separate the sound with its frequency and amplitude accordingly. However, for a common spectrogram, it has a linear scale upon frequencies and amplitudes, while human beings listen on a logarithmic scale. For example, it is easy for us to tell the difference between a 100 Hz sound and a 200 Hz sound, while it is hard to tell the difference between 10000 Hz and 10100 Hz, even though they both possess a difference of 100 Hz. The reason why we adopt Decibel scale is similar; an amplitude of 0.1 and 1 has only 10 dB in difference, while they are separated far in amplitude format. In this project, we leveraged function in torchaudio to convert WAV file to Mel Spectrogram, where frequencies is filtered with Mel scale and amplitude is filtered with decibel scale. These two scales are adopted so that the voices fit how human beings' hearing system actually work.

頻譜圖是將聲音轉換為圖像的方法。他會通過傅立葉變換把聲音按照頻率和振幅進行分離。然而，對於普通的頻譜圖而言，



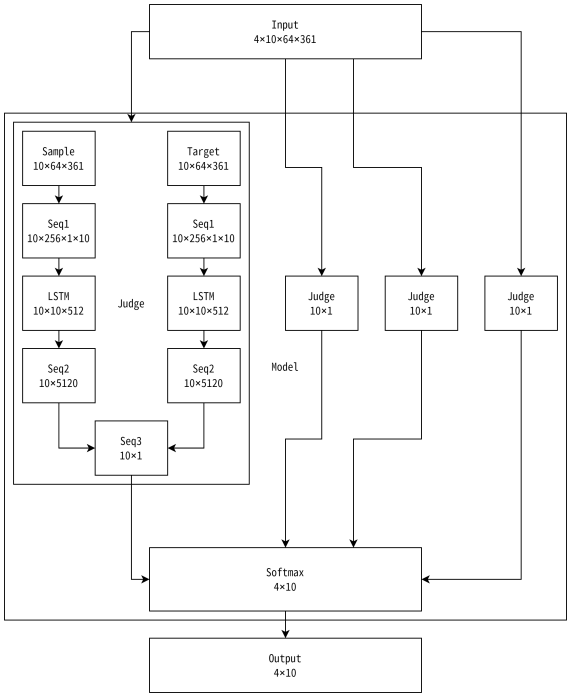
Above is the visualization of Mel Spectrogram. The x-axis represents timestamp in linear form and y-axis represents frequency in logarithmic form. The color of each coordinate of form (time, frequency) is obtained by converting amplitude into decibel format, which is logarithmic.

它在頻率和振幅上採用的是線性的，但是人類對於聲音的感受是對數關係的。比方說我們可以輕鬆地分辨出 100 Hz 聲音和 200 Hz 聲音之間的差異，但卻很難分辨出 10000 Hz 和 10100 Hz 之間的差異。我們判斷聲音的大小也是透過分貝的大小來進行判斷，像是振幅 0.1 和 1 之間雖然只有 10 分貝的差異，但在振幅格式上卻有很大的差距。在我們的 Final Project 中，我們利用了 torchaudio 中的函數將 WAV 文件轉換為 Mel Spectrogram，其中頻率使用 Mel scale 轉換，振幅則是轉換成分貝。採用這兩種刻度是為了讓聲音能夠更加符合人類的聽覺實際運作方式。

Mel Spectrogram is used to solve the extremity problems of the amplitude.

Model Diagram

The numbers on the diagram represent the sizes of the outputs at each layer. In this example, we found that it is the best case scenario. The input for this example is a batch of 2D Mel spectrograms. Each case in the batch is individually passed to the judge for similarity calculation. Although there are four judges shown in the diagram, in practice, we only use a single instance of the judge. The four judges in the diagram are included for ease of understanding. The judge consists of four stages, each stage containing multiple layers, with only one layer of LSTM. Each input batch is split into samples and targets, which are sequentially passed through seq1, LSTM, and seq2. Finally, the outputs of the sample and target are combined and passed to seq3, which represents the output of the judge. All the outputs of the judges are then concatenated and fed into Softmax to obtain the final output of the model, representing the probability distribution of the model's prediction of the identity of the sound.



In the diagram, seq1 contains various layers related to convolution, such as the convolution itself, max pooling, and ReLU. Seq2 flattens the output of seq1 after passing through LSTM. Seq3 is a discriminator composed of multiple DNNs and ReLU activations, which outputs a similarity score ranging from 0 to 1.

Next, I will discuss the parts of our model that can be adjusted, which are the aspects we modified during our experiments.

- The input can be either a 2D Mel spectrogram or a 1D amplitude sequence.
- In Seq1, the choice between using Conv1d or Conv2d depends on the input.
- The LSTM shown in the diagram can be replaced with a GRU, depending on the requirements.
- If RNN is not used, there will be no LSTM and Seq2 in the diagram. Seq3 will be adjusted to become the new Seq2.

圖上的數字代表著該層輸出的大小，這個例子是我們在最後發現是最好的case，他的input是有batch的2d Mel spectrogram。接著batch中的每一個case會分別丟到judge計

算相似度，圖中雖然有4個judge，但是實際上實作的時候只有使用一個judge的instance，圖上有4個judge只是為了方便理解。judge裡面會有四個階段，每個seq階段裡面很多個layer，而LSTM只有一層layer。每個input的batch拆分開來會有sample跟target，依序丟到seq1、LSTM、seq2裡面。最後再將sample跟target的輸出一起丟到seq3裡面，就是judge的輸出。接著將所有的judge輸出連接後傳入softmax裡面，就可以得到模型的最終輸出，代表著模型猜測這個聲音是誰的機率分布。

其中上圖的seq1裡面包含了convolution的相關layer，像是convolution本身、maxpooling、以及ReLU等，seq2裡面則是將seq1經過LSTM的輸出攤平，seq3則是一個由很多個DNN和ReLU結合起來的鑑別器，他會輸出一個介於0到1之間代表著相似度的數字。

接下來是有關我們模型可以調整的部分，也就是我們做實驗測試哪個部分能夠讓模型更好有修改到的部分。

- 輸入可以選2d mel spectrogram或是1d的震幅序列
- Seq1裡面可以根據輸入來選擇是用Conv1d或是Conv2d
- 上圖的LSTM也可根據需求改成GRU
- 如果沒有用RNN的話，就不會有上圖的LSTM和Seq2，Seq3會遞補成新的Seq2

Experiments

Training steps

Generally, to avoid overfitting, we should adjust the total training steps according to the quantity and characteristic of data and model. We conduct experiment on different training steps. As the training resources are limited, we adopt 10000 steps after finding that further increasing training steps would not improve the accuracy and might even lead to overfitting.

通常，為了避免過擬合，我們會需要根據資料的數量和特性以及模型進行調整總共的訓練步驟。我們有對不同的訓練步驟進行了實驗。但是由於訓練資源有限，我們發現進一步增加訓練步驟不會提高準確率，甚至可能導致過擬合，所以我們之後的時候都選擇採用了 10000 步的訓練步數。

With RNN vs Without RNN

RNN is used to remember previous or bidirectional information so that the model can consider these factors later on. Compared to architectures that do not use RNN, using an RNN architecture requires longer training time. In the subsequent analysis of the results, we will examine whether incorporating RNN has an impact on the final model performance.

RNN 是用來記憶先前或雙向的資訊，以便模型之後可以考慮這些因素。跟沒有使用 RNN 的架構相比之下，使用 RNN 架構會需要更長的訓練時間。在接下來的結果分析中，我們會分析加入 RNN 是否會對最終的模型表現造成影響。

LSTM vs GRU

LSTM and GRU have different architectural characteristics and convergence behaviors. LSTM has a more complex architecture and is better suited for long-term memory retention, but it may take longer time to converge. On the other hand, GRU has a simpler architecture and converges faster, requiring fewer steps and less data.

LSTM 和 GRU 在結構的複雜度和收斂的速度上有所不同。LSTM 有更複雜的結構，比較適合長期記憶保留，但可能需要比較長的時間才能收斂。相比之下，GRU 具有較簡單的結構，收斂速度更快，而且需要步數和資料更少。

1 layer RNN vs 2 layer RNN

Generally, a model using RNN with 2 layers would possess more complexity than the one with 1 layer. In this case, 19890369 for 2 layers and 18313409 for 1 layer. In the conduction of our experiment, we'll compare the performance of 1 layer and 2 layer, and modify the total training steps in 2 layer model.

一般而言，使用兩層 RNN 的模型比使用一層的模型更複雜。在本專案中，兩層的模型有 19890369 個參數，而一層的模型則有 18313409 個參數。此外，我們將比較一層和兩層模型的性能。

Conv1d with original voice vs Conv2d with Mel Spectrogram

According to this article, since human beings hear voice in a logarithmic scale rather than a linear scale, the difference between 100 Hz and 200 Hz is more significant than that of 10000 Hz and 10100 Hz, amplitude similar. To fix the problem that scalar difference may differ from hearing difference, Mel Spectrogram can be used to fix the problem on both sound frequency and amplitude. In the following part, we conduct experiment on sound converted to Mel Spectrogram and original data, and compare their performances.

根據這篇文章，由於人類聽到的聲音是以對數尺度而不是線性尺度來感知的，所以 100 Hz 和 200 Hz 之間的差異比 10000 Hz 和 10100 Hz 之間的差異更明顯，即使振幅相似。為了解決標量差異可能與聽覺差異不一致的問題，可以使用 Mel Spectrogram 來修正聲音頻率和振幅上的問題。在接下來的部分，我們將對轉換為 Mel Spectrogram 和原始數據的聲音進行實驗，並比較它們的性能表現。

Evaluation metrics

原始資料集 模型表現

遞迴類型	卷積類型	Aa 名稱	↗ 10 人實驗數據	# 10 人準確率	↗ 6 人實驗數據	# 6 人準確率	有無 RNN	RNN 層數
None	Conv1d	Conv1d without RNN	實驗 10-6	1	實驗 6-5	2	without RNN	None
GRU	Conv1d	Conv1d with 1 layer GRU	實驗 10-1	6	實驗 6-11	5	with RNN	RNN layer=1
GRU	Conv1d	Conv1d with 2 layer GRU	實驗 10-7	3	實驗 6-15	2	with RNN	RNN layer=2
LSTM	Conv1d	Conv1d with 1 layer LSTM	實驗 10-2	7	實驗 6-2	5	with RNN	RNN layer=1
LSTM	Conv1d	Conv1d with 2 layer LSTM	實驗 10-3	7	實驗 6-12	5	with RNN	RNN layer=2
None	Conv2d	Conv2d without RNN	實驗 10-8	0	實驗 6-3	1	without RNN	None
GRU	Conv2d	Conv2d with 1 layer GRU	實驗 10-9	7	實驗 6-13	5	with RNN	RNN layer=1
GRU	Conv2d	Conv2d with 2 layer GRU	實驗 10-10	6	實驗 6-14	5	with RNN	RNN layer=2
LSTM	Conv2d	Conv2d with 1 layer LSTM	實驗 10-5	5	實驗 6-4	6	with RNN	RNN layer=1
LSTM	Conv2d	Conv2d with 2 layer LSTM	實驗 10-11	7	實驗 6-9	6	with RNN	RNN layer=2

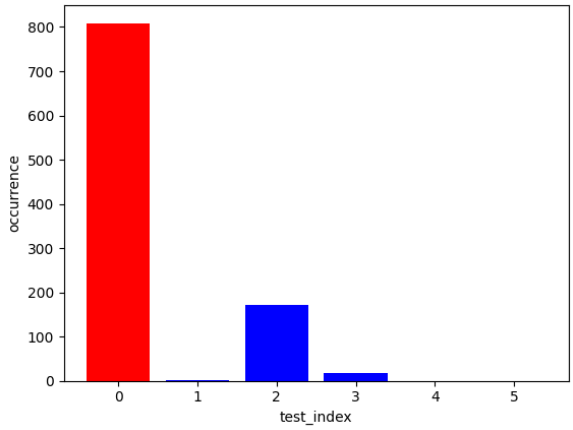
人聲資料集（6人） 模型表現

Aa 名稱	↗ 2 真人實驗數據	# 2 真人準確率	↗ 1 真人實驗數據	# 1 真人準確率
Conv1d with 1 layer GRU	實驗 6-17 (2 real)	1	實驗 6-18 (1 real)	6
Conv1d with 1 layer LSTM	實驗 6-7 (2 real)	3	實驗 6-6 (1 real)	5
Conv2d with 1 layer GRU	實驗 6-16 (2 real)	5	實驗 6-19 (1 real)	5
Conv2d with 1 layer LSTM	實驗 6-8 (2 real)	5	實驗 6-20 (1 real)	5

Validation

Below is the introduction to how we validate the accuracy of the model. To avoid possible biased output that may happen when running insufficient times, we run the same voice on model repeatedly. For each person that the model can specify, we run 1000 times on the voice as input of model for validation. Among all the outputs of these 1000 tests, argmax is leveraged to vote for the most convincing consequence, and set as output.

Our model will ultimately output a graph like this. The "occurrence" on the graph represents the occurrences of the model's predictions for this person's voice among the 1000 outputs. The "test index" indicates the total number of individuals, which in this case is 6. In this graph, it shows that there are 6 individuals.



As for the colors, red represents the person to whom the voice segment belongs, while blue represents voices that do not belong to that person. Therefore, in this graph, it indicates that the model's prediction is correct. The highest voted result by the model is index 0, which corresponds to the red color, representing the true voice owner. This means that in this particular test, the model was able to correctly predict the result.

以下為我們驗證模型準確性的方法介紹。為了盡量避免因執行次數不足而產生的輸出誤差，我們會重複在模型上輸入相同的語音。對於每個能被模型識別出的人，我們會把他的語音重複輸入模型1000次以進行驗證。最後，我們從這1000次測試的輸出中，利用argmax來投票選出最有說服力的結果，並將其輸出。

舉例來說，我們的模型最終會輸出這種圖片，圖上的 occurrence 代表著這1000次的輸出中，模型輸出所猜測是這個人的聲音的情形，test index 則代表著總共有幾個人，在這張圖的情況代表著有 6 個人。至於顏色的部分，紅色代表著這個聲音片段是誰的聲音，藍色的則代表不是這個人的聲音。所以像是這張圖片代表著模型預測正確，因為模型投票出來的結果最高的是 index 0，而紅色，也就是真正的聲音主人也是 index 0，所以就代表著這個在這次的測試中，模型能夠正確地預測出結果。

Results & Analysis & Others

10 人實驗數據

Aa 名稱	卷積類型	遞迴類型	# 準確率	筆記本	# 步數	# 學習率	# 權重衰退
實驗 10-1	Conv1d	GRU layer=1	6	Conv1d with RNN.ipynb	10000	0.00003	0.00001
實驗 10-2	Conv1d	LSTM layer=1	7	Conv1d with RNN.ipynb	10000	0.00003	0.00001
實驗 10-3	Conv1d	LSTM layer=2	7	Conv1d with RNN.ipynb	10000	0.00003	0.00001
實驗 10-4	Conv1d	LSTM layer=2	7	Conv1d with RNN.ipynb	50000	0.00003	0.00001
實驗 10-5	Conv2d	LSTM layer=1	5	MelSpectrogram with RNN.ipynb	10000	0.00003	0.00001
實驗 10-6	Conv1d	None	1	Conv1d without RNN.ipynb	10000	0.00003	0.00001
實驗 10-7	Conv1d	GRU layer=2	3	Conv1d with RNN.ipynb	10000	0.00003	0.00001
實驗 10-8	Conv2d	None	0	MelSpectrogram without RNN.ipynb	10000	0.00003	0.00001
實驗 10-9	Conv2d	GRU layer=1	7	MelSpectrogram with RNN.ipynb	10000	0.00003	0.00001
實驗 10-10	Conv2d	GRU layer=2	6	MelSpectrogram with RNN.ipynb	10000	0.00003	0.00001
實驗 10-11	Conv2d	LSTM layer=2	7	MelSpectrogram with RNN.ipynb	10000	0.00003	0.00001

6 人實驗數據

Aa 名稱	卷積類型	遞迴類型	# 準確率	筆記本	# 步數	# 學習率	# 權重衰退
實驗 6-1	Conv1d	LSTM layer=1	6	Conv1d with RNN.ipynb	10000	0.00003	0
實驗 6-2	Conv1d	LSTM layer=1	5	Conv1d with RNN.ipynb	10000	0.00003	0.00001
實驗 6-3	Conv2d	None	1	MelSpectrogram without RNN.ipynb	10000	0.00003	0.00001
實驗 6-4	Conv2d	LSTM layer=1	6	MelSpectrogram with RNN.ipynb	10000	0.00003	0.00001
實驗 6-5	Conv1d	None	2	Conv1d without RNN.ipynb	10000	0.00003	0.00001
實驗 6-6 (1 real)	Conv1d	LSTM layer=1	5	Conv1d with RNN with human.ipynb Conv1d with RNN with human correct version.ipynb	10000	0.00003	0.00001
實驗 6-7 (2 real)	Conv1d	LSTM layer=1	3	Conv1d with RNN.ipynb Conv1d with RNN with human.ipynb	10000	0.00003	0.00001
實驗 6-8 (2 real)	Conv2d	LSTM layer=1	5	MelSpectrogram with RNN with human.ipynb	10000	0.00003	0.00001
實驗 6-9	Conv2d	LSTM layer=2	6	MelSpectrogram with RNN.ipynb	10000	0.00003	0.00001
實驗 6-10 (2 real)	Conv2d	LSTM layer=2	5	MelSpectrogram with RNN with human.ipynb	10000	0.00003	0.00001
實驗 6-11	Conv1d	GRU layer=1	5	Conv1d with RNN.ipynb	10000	0.00003	0.00001
實驗 6-12	Conv1d	LSTM layer=2	5	Conv1d with RNN.ipynb	10000	0.00003	0.00001
實驗 6-13	Conv2d	GRU layer=1	5	MelSpectrogram with RNN.ipynb	10000	0.00003	0.00001
實驗 6-14	Conv2d	GRU layer=2	5	MelSpectrogram with RNN.ipynb	10000	0.00003	0.00001
實驗 6-15	Conv1d	GRU layer=2	2	Conv1d with RNN.ipynb	10000	0.00003	0.00001
實驗 6-16 (2 real)	Conv2d	GRU layer=1	5	MelSpectrogram with RNN with human.ipynb	10000	0.00003	0.00001
實驗 6-17 (2 real)	Conv1d	GRU layer=1	1	Conv1d with RNN with human.ipynb	10000	0.00003	0.00001
實驗 6-18 (1 real)	Conv1d	GRU layer=1	6	Conv1d with RNN with human.ipynb	10000	0.00003	0.00001
實驗 6-19 (1 real)	Conv2d	GRU layer=1	5	MelSpectrogram with RNN with human.ipynb	10000	0.00003	0.00001
實驗 6-20 (1 real)	Conv2d	LSTM layer=1	5	MelSpectrogram with RNN with human.ipynb	10000	0.00003	0.00001

Type of experiment

Training steps

Through the comparison of experiments 10-3 and 10-4, we observed that despite the significantly fewer training steps in experiment 10-3, the results were comparable to experiment 10-4. This finding suggests that reducing the number of training steps does not have a significant impact on the final performance of the model. This not only helps save computational resources but also mitigates the risk of overfitting. Therefore, we have set the number of training steps to 10,000 for subsequent experiments.

It is crucial to strike a balance between model training and the data available to avoid both overfitting and underfitting. By maintaining this balance, we can mitigate the risk of overfitting while ensuring that the model learns effectively from the data. This approach not only ensures the integrity of our model's performance but also enables us to allocate computational resources more efficiently.

透過比較實驗10-3和實驗10-4，雖然實驗10-3的訓練步數比實驗10-4少了很多，但是我們可以觀察到實驗10-3的結果和實驗10-4的結果差不多，並不會對模型的最終結果有顯著的影響。這個發現除了可以替我們節省運算資源，也可以避免模型產生過擬合。所以我們之後的訓練步數都是設在10000步。在模型的訓練和資料集的數據之間保持平衡，不要讓他過擬合也不要讓他完全學不到東西。如此一來，我們可以減輕過度擬合的風險，並且能夠更好的利用資源。這種方法不僅確保了我們模型成果的完整性，還確保我們能夠有效分配計算資源。

With RNN vs Without RNN

原始資料集 模型表現

Aa 名稱	↗ 10 人實驗數據	# 10 人準確率	↗ 6 人實驗數據	# 6 人準確率	遞迴類型	卷積類型	RNN 層數	有無 RNN
Conv1d without RNN	實驗 10-6	1	實驗 6-5	2	None	Conv1d	None	without RNN
Conv1d with 1 layer GRU	實驗 10-1	6	實驗 6-11	5	GRU	Conv1d	RNN layer=1	with RNN
Conv1d with 2 layer GRU	實驗 10-7	3	實驗 6-15	2	GRU	Conv1d	RNN layer=2	with RNN
Conv1d with 1 layer LSTM	實驗 10-2	7	實驗 6-2	5	LSTM	Conv1d	RNN layer=1	with RNN
Conv1d with 2 layer LSTM	實驗 10-3	7	實驗 6-12	5	LSTM	Conv1d	RNN layer=2	with RNN
Conv2d without RNN	實驗 10-8	0	實驗 6-3	1	None	Conv2d	None	without RNN
Conv2d with 1 layer GRU	實驗 10-9	7	實驗 6-13	5	GRU	Conv2d	RNN layer=1	with RNN
Conv2d with 2 layer GRU	實驗 10-10	6	實驗 6-14	5	GRU	Conv2d	RNN layer=2	with RNN
Conv2d with 1 layer LSTM	實驗 10-5	5	實驗 6-4	6	LSTM	Conv2d	RNN layer=1	with RNN
Conv2d with 2 layer LSTM	實驗 10-11	7	實驗 6-9	6	LSTM	Conv2d	RNN layer=2	with RNN

Based on the comparison between experiments that include and exclude the use of RNN, we can observe that models incorporating RNN tend to yield better results compared to models without RNN. This observation is evident in the 6-person experiments, where experiments 6-3 and 6-5 do not utilize RNN, while the rest of the experiments involving 6 individuals employ RNN. Similarly, in the 10-person experiments, experiments 10-6 and 10-8 do not use RNN, while the others do.

The results presented in the table indicate that the inclusion of RNN in the model architecture has a positive impact on performance. This finding suggests that exploring other methods or architectures that involve RNN could potentially further enhance the performance of our model. Therefore, in the upcoming stages, our focus will primarily be on training models that incorporate RNN.

接下來是比較有沒有使用 RNN 對結果好壞影響的差異。像是在六人的實驗中，實驗 6-3 和 6-5 沒有使用到 RNN，除此之外6人的實驗中都有使用到 RNN。而10人的部分，實驗10-6以及10-8沒有使用到RNN，其他則都有使用到。從表格來看，我們可以發現有使用 RNN 的模型效果會比沒有使用 RNN 的模型效果還要好，這個觀察結果代表我們需要去尋找其他有使用到RNN的方法或架構來讓我們的模型表現更好。因此，在接下來的部分，我們將主要使用有 RNN 的模型進行訓練。

1 layer RNN vs 2 layer RNN

原始資料集 模型表現

Aa 名稱	↗ 10 人實驗數據	# 10 人準確率	↗ 6 人實驗數據	# 6 人準確率	遞迴類型	卷積類型	有無 RNN	RNN 層數
Conv1d without RNN	實驗 10-6	1	實驗 6-5	2	None	Conv1d	without RNN	None
Conv1d with 1 layer GRU	實驗 10-1	6	實驗 6-11	5	GRU	Conv1d	with RNN	RNN layer=1
Conv1d with 2 layer GRU	實驗 10-7	3	實驗 6-15	2	GRU	Conv1d	with RNN	RNN layer=2
Conv1d with 1 layer LSTM	實驗 10-2	7	實驗 6-2	5	LSTM	Conv1d	with RNN	RNN layer=1
Conv1d with 2 layer LSTM	實驗 10-3	7	實驗 6-12	5	LSTM	Conv1d	with RNN	RNN layer=2
Conv2d without RNN	實驗 10-8	0	實驗 6-3	1	None	Conv2d	without RNN	None

Aa 名稱	↗ 10 人實驗數據	# 10 人準確率	↗ 6 人實驗數據	# 6 人準確率	遞迴類型	卷積類型	有無 RNN	RNN 層數
Conv2d with 1 layer GRU	實驗 10-9	7	實驗 6-13	5	GRU	Conv2d	with RNN	RNN layer=1
Conv2d with 2 layer GRU	實驗 10-10	6	實驗 6-14	5	GRU	Conv2d	with RNN	RNN layer=2
Conv2d with 1 layer LSTM	實驗 10-5	5	實驗 6-4	6	LSTM	Conv2d	with RNN	RNN layer=1
Conv2d with 2 layer LSTM	實驗 10-11	7	實驗 6-9	6	LSTM	Conv2d	with RNN	RNN layer=2

During our experiments with different models and architectures, we observed some interesting phenomena. For LSTM, experiments 6-2 and 6-12, which used Conv1d without real human voice, showed similar performance, with an accuracy of correctly identifying 5 out of 6 individuals. Similarly, in the case of using Conv2d with two real human voices, experiments 6-8 and 6-10 had similar results, with an accuracy of correctly identifying 5 out of 6 individuals. On the other hand, when using Conv2d without real human voice, experiments 6-4 and 6-9 exhibited similar performance, with each experiment achieving an accuracy of correctly identifying all 6 individuals.

In the case of GRU models, experiments 6-13 and 6-14, which used Conv2d without real human voice, achieved an accuracy of correctly identifying 5 out of 6 individuals. Furthermore, experiments 6-11 and 6-15, which used Conv1d without real human voice, had accuracies of correctly identifying 5 and 2 individuals, respectively.

LSTM and GRU models performed similarly overall, but in the case of using Conv1d without real human voice, the performance of the 2-layer GRU model was not as good as the 1-layer GRU model. Based on an article from Stack Overflow, we speculate that this difference may be due to the tendency of multi-layer GRU models to overfit. Additionally, Conv1d might have difficulty extracting meaningful features, which could contribute to the lower performance of the 2-layer GRU model with Conv1d. Therefore, considering that the 1-layer model has faster training speed and the 2-layer model does not offer better accuracy, the subsequent discussion will primarily focus on the 1-layer model.

我們在使用不同的模型和架構進行實驗時，發現了一些有趣的現象。像是就 LSTM 而言，使用 Conv1d 而且沒有真人聲音的實驗 6-2 和實驗 6-12 的表現相似，正確率都是 6 人能正確辨識 5人。同樣地，對於使用 Conv2d 且有使用 2 個真人聲音的情況，有使用2個真人聲音實驗 6-8和實驗 6-10的結果也相近，正確率都是 6 人能正確辨識 5人。另一方面，當使用 Conv2d 且沒有真實人聲時，實驗 6-4 和實驗 6-9 都有相似的表現，每個實驗正確率都是 6 人能正確辨識 6人。
至於在 GRU 模型中，使用 Conv2d 且沒有真人聲音的實驗 6-13 和實驗 6-14 正確率都是 6 人能正確辨識 5人。此外，使用 Conv1d 且沒有真實人聲的實驗 6-11 和實驗 6-15 正確率分別是 6 人能正確辨識 5人和 2人。
LSTM 和 GRU 兩組表現大致相同，但是在使用 Conv1d 且沒有真實人聲時，2 層 GRU 模型的表現卻不如 1 層 GRU 模型。根據一篇來自Stack Overflow的文章，我們猜測這種差異可能是因為多層的 GRU 模型容易過擬合所導致的。此外，Conv1d 可能也比較難提取有意義的特徵，因此這兩點就導致了2 layer GRU使用 Conv1d 會有較差的表現。所以考慮到1層的模型訓練速度較快且 2 層模型並沒有較好的準確度，後續的討論將主要集中在 1 層模型上。

LSTM vs GRU

原始資料集 模型表現

Aa 名稱	↗ 10 人實驗數據	# 10 人準確率	↗ 6 人實驗數據	# 6 人準確率	遞迴類型	卷積類型	有無 RNN	RNN 層數
Conv1d without RNN	實驗 10-6	1	實驗 6-5	2	None	Conv1d	without RNN	None
Conv1d with 1 layer GRU	實驗 10-1	6	實驗 6-11	5	GRU	Conv1d	with RNN	RNN layer=1
Conv1d with 2 layer GRU	實驗 10-7	3	實驗 6-15	2	GRU	Conv1d	with RNN	RNN layer=2
Conv1d with 1 layer LSTM	實驗 10-2	7	實驗 6-2	5	LSTM	Conv1d	with RNN	RNN layer=1
Conv1d with 2 layer LSTM	實驗 10-3	7	實驗 6-12	5	LSTM	Conv1d	with RNN	RNN layer=2
Conv2d without RNN	實驗 10-8	0	實驗 6-3	1	None	Conv2d	without RNN	None
Conv2d with 1 layer GRU	實驗 10-9	7	實驗 6-13	5	GRU	Conv2d	with RNN	RNN layer=1

Aa 名稱	↗ 10 人實驗數據	# 10 人準確率	↗ 6 人實驗數據	# 6 人準確率	遞迴類型	卷積類型	有無 RNN	RNN 層數
Conv2d with 2 layer GRU	實驗 10-10	6	實驗 6-14	5	GRU	Conv2d	with RNN	RNN layer=2
Conv2d with 1 layer LSTM	實驗 10-5	5	實驗 6-4	6	LSTM	Conv2d	with RNN	RNN layer=1
Conv2d with 2 layer LSTM	實驗 10-11	7	實驗 6-9	6	LSTM	Conv2d	with RNN	RNN layer=2

We can observe that the GRU model converges faster than the LSTM model when there are 10 individuals. Additionally, in terms of accuracy, GRU generally performs similarly to LSTM or slightly worse. However, in the case of a 1-layer Conv2d, the experiment with a GRU of 10-9 performs better than the LSTM result of 10-5. We speculate that this could be due to unlucky training circumstances.

In the scenario of 6 individuals without real human voice, both GRU and LSTM achieve similar results, except for the 2-layer Conv1d GRU, which performs significantly worse than LSTM. The accuracies are 2 individuals correctly identified out of 6 and 5 individuals correctly identified out of 6, respectively. We speculate that this could be due to unforeseen circumstances, such as particularly poor training luck, or the training steps for the 2-layer GRU should not have been set to 10,000 steps. Its fast convergence may have led to overfitting, resulting in poorer final results.

Lastly, in the case of 6 individuals with real human voice, when there is only 1 real human voice, LSTM and GRU achieve similar results. However, when there are 2 real human voices, it can be observed that using Conv2d with GRU and LSTM performs significantly better than using Conv1d with GRU and LSTM. We speculate that this could be because human voices are more unpredictable than Bark, and therefore, using Conv2d for transformation allows the models to capture features better and achieve more accurate recognition.

我們可以觀察到在 10 人的時候， GRU 模型的收斂速度比 LSTM 模型快。另外就準確度而言，GRU 的表現通常和 LSTM 相同或是稍微差一點。不過在 1 layer Conv2d 的時候，實驗 10-9 的 GRU 卻能夠比實驗 10-5 的 LSTM 結果還要好，我們猜測有可能是因為訓練的運氣不好。

至於在 6 人而且沒有真人聲音的情況下，GRU以及LSTM的成果大致相等，但是實驗6-15比實驗6-12的成果還要差非常多，分別是6個人有正確辨識到2人以及6個人有正確辨識到5人。我們猜測有可能是因為意外狀況，像是訓練的運氣特別差，或是2 layer GRU設置的訓練步數不該設置成10000步，因為他會快速收斂，可能太快就導致過擬合的情形，導致最終結果較差。

最後，在6人且有真人聲音的情況下，在只有1個真人聲音的時候LSTM跟GRU的成果也都大致相等，但是在2個真人聲音的情況時，可以發現使用Conv2d的GRU以及LSTM能夠比Conv1d的GRU以及LSTM還要好很多，我們猜測這個原因是因為震幅是一個數字，所以他會對數量級很敏感，但是Mel Spectrogram 卻不會不會因為數量級而受到大幅變動，因此要使用Conv2d進行轉換，模型才能夠更好的抓到特徵，進而正確的辨識。所以接下來我會開始分析在使用Conv1d以及Conv2d對於模型表現的影響。

Conv1d with original voice vs Conv2d with Mel Spectrogram

原始資料集 模型表現

Aa 名稱	↗ 10 人實驗數據	# 10 人準確率	↗ 6 人實驗數據	# 6 人準確率	遞迴類型	卷積類型	RNN 層數	有無 RNN
Conv1d without RNN	實驗 10-6	1	實驗 6-5	2	None	Conv1d	None	without RNN
Conv1d with 1 layer GRU	實驗 10-1	6	實驗 6-11	5	GRU	Conv1d	RNN layer=1	with RNN
Conv1d with 2 layer GRU	實驗 10-7	3	實驗 6-15	2	GRU	Conv1d	RNN layer=2	with RNN
Conv1d with 1 layer LSTM	實驗 10-2	7	實驗 6-2	5	LSTM	Conv1d	RNN layer=1	with RNN
Conv1d with 2 layer LSTM	實驗 10-3	7	實驗 6-12	5	LSTM	Conv1d	RNN layer=2	with RNN
Conv2d without RNN	實驗 10-8	0	實驗 6-3	1	None	Conv2d	None	without RNN
Conv2d with 1 layer GRU	實驗 10-9	7	實驗 6-13	5	GRU	Conv2d	RNN layer=1	with RNN
Conv2d with 2 layer GRU	實驗 10-10	6	實驗 6-14	5	GRU	Conv2d	RNN layer=2	with RNN

Aa 名稱	↗ 10 人實驗數據	# 10 人準確率	↗ 6 人實驗數據	# 6 人準確率	遞迴類型	卷積類型	RNN 層數	有無 RNN
Conv2d with 1 layer LSTM	實驗 10-5	5	實驗 6-4	6	LSTM	Conv2d	RNN layer=1	with RNN
Conv2d with 2 layer LSTM	實驗 10-11	7	實驗 6-9	6	LSTM	Conv2d	RNN layer=2	with RNN

We can observe that, with the exception of experiments 10-6 and 10-8, which performed poorly due to the lack of RNN usage, the remaining experiments show decent performance when there are 10 individuals. However, in the case of a 2-layer GRU, we notice that experiment 10-7 performs significantly worse than experiment 10-10, correctly identifying only 3 individuals compared to 6 individuals. This discrepancy could be attributed to the potential overfitting issue caused by Conv1d.

In the scenario of 6 individuals without real human voices, the performance is generally comparable across the experiments. However, in the case of a 2-layer GRU, experiment 6-15 performs much worse than experiment 6-14, correctly identifying only 2 individuals instead of 5. This could also be attributed to the aforementioned issue of overfitting in the 2-layer GRU.

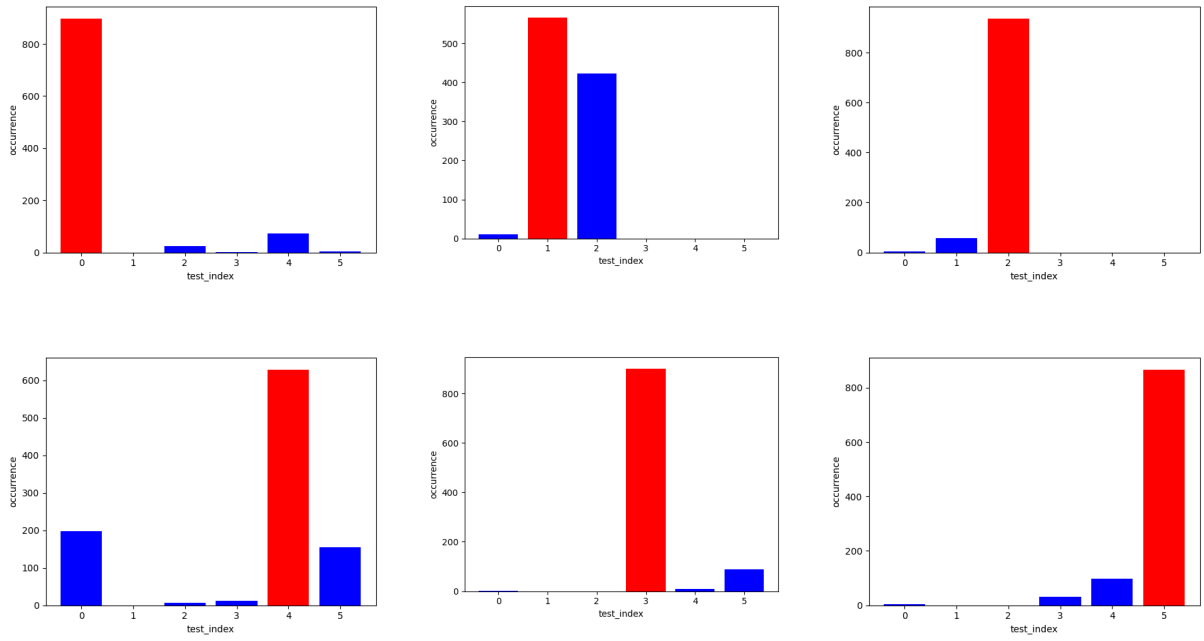
Lastly, in the scenario of 6 individuals with real human voices, we observe that the results of Conv2d are consistently accurate, correctly identifying 5 individuals. On the other hand, the results of Conv1d are highly inconsistent, ranging from 1 to 6 individuals correctly identified. This further confirms the earlier mentioned observation that using Conv2d for feature extraction enables the model to better capture features and consequently make accurate identifications.

我們可以觀察到在 10 人的時候， 除了實驗 10-6 以及實驗 10-8 是因為沒有使用RNN所以導致結果很差以外，其他都能夠有不錯的表現。但是在 2 layer GRU 中，可以發現實驗 10-7 比實驗 10-10 的結果還要差很多，分別是 10 個裡面正確辨識 3 人以及 6 人，我們認為可能是 Conv1d 會太容易 overfitting 所導致的。

至於在 6 人而且沒有真人聲音的情況下，表現皆大致相等，但是在 2 layer GRU 的情況底下，實驗 6-15 卻比實驗 6-14 還要差很多，分別是正確辨識 2 人以及 5 人。可能就是因為前面提到的，2 layer GRU 更容易 overfitting 所導致的。

最後，在 6 人且有真人聲音的情況下，可以發現 Conv2d 的結果大致上都非常穩定的落在正確辨識 5 人，但是 Conv1d 的結果卻非常不穩定，從 1 到 6 都有。這也驗證了我剛剛所提到的，要使用 Conv2d 進行轉換，模型才能夠更好的抓到特徵，進而正確的辨識。

Best result for future practical usages



The best experimental results we achieved on the dataset containing human voice were obtained using a model trained with Conv2d and a single layer LSTM, specifically the results of experiment 6-4. Additionally, in this model, we obtained consistent results with previous inferences.

Firstly, we used Conv2d instead of Conv1d due to the inclusion of human voice in the dataset. This choice was made to avoid the scaling issue caused by the different formats of audio files.

Secondly, we opted for a single layer LSTM instead of a two-layer LSTM because a single layer LSTM incurs lower computational costs, effectively saving training and inference time as well as computational resources. Furthermore, we chose a single layer LSTM over a single layer GRU because the GRU model tends to exhibit overfitting phenomena in training scenarios with 10,000 steps, a learning rate of 3e-5, and weight decay of 1e-5.

我們在包含真人聲音的資料集上，所取得的最佳實驗成果是基於 Conv2d 和 1 layer LSTM 訓練出的模型，也就是實驗 6-4 的結果。另外在此模型中，我們得出了和先前的推論一致的結果。

首先，由於資料集中包含真人的聲音，因此使用 Conv2d 而非 Conv1d，以規避因為聲音檔的格式不同而導致的數量級問題。

其次，使用 1 layer 的 LSTM 而非 2 layer 是因為 1 layer 的 LSTM 運算所需成本較低，可以有效地節約訓練和推論的時間和運算資源。此外，之所以使用 1 layer 的 LSTM 而非 1 layer 的 GRU，則是因為 GRU 的模型在 10000 步、3e-5 的學習率和 1e-5 的權重衰退的訓練情境下，較容易出現過擬合的現象。

Limitation of our work

1. Limitation of training

We have limited computational resources and time constraints. Although we have access to a GPU workstation provided by our instructor, we encountered difficulties in successfully training models on it. As a result, we conducted training on our local machines. Consequently, training a more complex model architecture like the 2 layer LSTM requires more time, but the improvement in performance is not significant compared to the 1 layer LSTM. Therefore, we decided not to run the real human testing scenario with the 2 layer LSTM as it is not feasible within our limitations.

我們的計算資源跟時間有限，雖然有老師提供的 GPU 工作站，但是我們不知道為何沒辦法成功在上面進行模型的訓練，因此在訓練的時候都是在我們的本地端電腦。所以像是 2 layer LSTM 在模型架構上比 1 layer 還要更複雜，需要花費更長的時間進行訓練，但是效果卻沒有比1 layer還要好多少，所以我們在真人資料集的實驗就沒有去跑 2 layer LSTM 的情形，因為太不切實際了。

2. Cannot specify when there are too many people

In the dataset, when testing with only two voices, the model achieves a high accuracy rate close to 100%. However, when there are six voices, the accuracy drops to 83.3%, and with ten voices, it further decreases to 70%. We suspect that this issue could be addressed by applying a model that can capture more distinctive audio features.

在資料集中如果只有測試兩個聲音的時候，模型的準確率可以接近 100%。但是當有 6 個聲音的時候，準確率卻會下降到 83.3%，當有 10 個聲音時，準確率更會下降到 70%。我們猜測這可以透過應用一個能對抓出更多聲音特徵的模型來解決這個問題。

3. Limitation on file formats

One limitation of our model is that it requires the use of normalized WAV files. During the experiments with real human voices, we discovered that the audio sources we used were not normalized WAV files. Since the voices generated by Bark were already in a normalized format, the presence of non-normalized voices severely affected the performance of our model. To ensure accuracy, it was necessary to convert the real human voices into mono-channel format and with a sampling rate of 24000 Hz.

Although we found methods to normalize the WAV files of the real human voices, it appears that this did not completely resolve the issue. However, when using Conv2D, the PyTorch library automatically handles the conversion of Mel Spectrograms, resulting in acceptable results. Nonetheless, it is essential to address this problem properly, potentially by considering alternative data sources with a more suitable format to ensure accurate performance.

我們模型的其中一個限制是需要使用經過正規化過後的的 WAV 文件。因為在進行真人聲音的實驗時，我們發現我們使用的聲音來源是一個沒有被正規化過的 WAV 文件。因為我們用 Bark 生成的聲音已經是正規化的形式，所以沒有被正規化的聲音會嚴重影響我們模型的性能。因此，為了保證準確性，需要將真人的聲音轉換為單聲道形式，並且具有 24000 Hz 的採樣率。雖然我們有找到方法讓真人聲音的WAV文件正規化，但是似乎不能完全解決這個問題，不過在使用Conv2d時，pytorch的函式庫會在轉換 Mel Spectrogram 的過程中幫我們解決這個問題，所以最後的結果並沒有太差，但是實際上應該需要解決這項問題，可能我們的資料來源需要更改，找到一個更適合的格式。

Approach to applying the model/method to practical use.

We trained our model using voice generated by Bark and tested its ability to recognize real human voices. We conducted experiments involving one real human paired with five voices generated by Bark, as well as two real humans paired with four voices generated by Bark. For these experiments, we only used models that performed well during training.

In the case of one real human, each model achieved good results. However, when there were two real humans involved, the models using Conv1D with 1 layer GRU and Conv1D with 1 layer LSTM did not perform as well.

我們透過Bark所生成出來的聲音訓練，並利用這個聲音模型識別真人聲音。我們只有測試加入一個真人配上五個由Bark生出來的假人以及兩個真人配上四個由Bark生出來的假人的實驗。我們都只有拿在訓練結果還不錯的模型來跑有真人的情形。在只有一個真人的情況底下，每個模型都能夠有不錯的結果，但是在兩個真人的情況底下，Conv1d with 1 layer GRU 和 Conv1d with 1 layer LSTM 的結果卻沒有非常好。

人聲資料集（6人） 模型表現

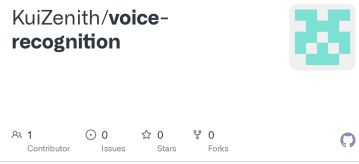
Aa 名稱	↗ 2 真人實驗數據	# 2 真人準確率	↗ 1 真人實驗數據	# 1 真人準確率
Conv1d with 1 layer GRU	實驗 6-17 (2 real)	1	實驗 6-18 (1 real)	6
Conv1d with 1 layer LSTM	實驗 6-7 (2 real)	3	實驗 6-6 (1 real)	5
Conv2d with 1 layer GRU	實驗 6-16 (2 real)	5	實驗 6-19 (1 real)	5
Conv2d with 1 layer LSTM	實驗 6-8 (2 real)	5	實驗 6-20 (1 real)	5

Reference

GitHub - KuiZenith/voice-recognition

Contribute to KuiZenith/voice-recognition development by creating an account on GitHub.

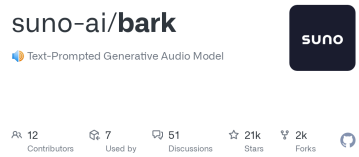
https://github.com/KuiZenith/voice-recognition



GitHub - suno-ai/bark: 🔊 Text-Prompted Generative Audio Model

🔊 Text-Prompted Generative Audio Model. Contribute to suno-ai/bark development by creating an account on GitHub.

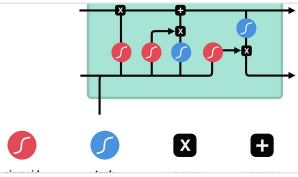
https://github.com/suno-ai/bark



25. 簡介LSTM 與GRU

RNN 容易有梯度消失與梯度爆炸的問題，而LSTM 與GRU就是在解決這樣的問題。

https://medium.com/programming-with-data/25-簡介lstm-與gru-3e0eaa100d29



Audio Deep Learning Made Simple (Part 2): Why Mel Spectrograms perform better

A Gentle Guide to processing audio in Python. What are Mel Spectrograms and how to generate them, in Plain English.

https://towardsdatascience.com/audio-deep-learning-made-simple-part-2-why-mel-spectrograms-perform-better-aad889a93505

Guardians of the Galaxy | Chris Pratt, Zoe Saldana & Vin Diesel | Talks at Google

Some of our very own guardians, Chris Pratt, Zoe Saldana, and Vin Diesel take time out of their duties protecting us from evil to swing by Google NYC to discuss "Guardians of the Galaxy," their roles within, what it's like to be action figures, and many other topics.

https://www.youtube.com/watch?v=YfQJed8PSLU&ab_channel=TalksatGoogle