Tomofun 狗音辨識 AI 百萬挑戰賽

初賽賽後報告

**隊伍：酒桃郎跟他的貓**

**成員：簡孝羽**

1. **環境**

作業系統: Windows 10 , RTX 3080 laptop, 64G RAM, 1TB SSD

語言 : Python 3.6

套件 : TensorFlow-gpu 2.5.0

Tensorflow\_addons

Liborsa

Scikit-learn

Pandas

Keras

Audiomentation

MixupGenerator

OpenCV

預訓練模型: EfficientNet B0

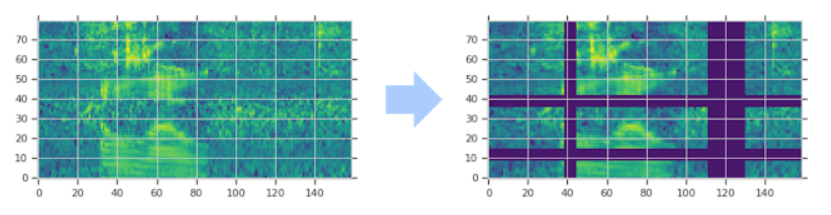
模型使用NoisyStudent權重 : <https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet>

額外資料集: 無

1. **作法摘要與介紹**

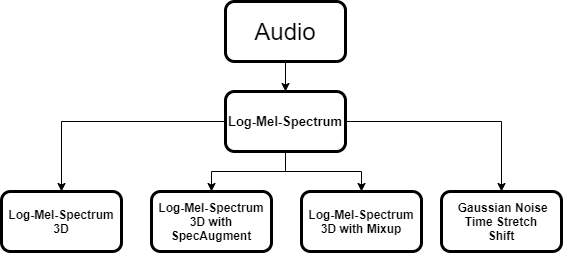
本次競賽本隊使用梅爾頻譜 (log-Mel-Spectrum)，以及梅爾頻譜的Delta Features 跟 Accelerate features 堆疊成三維影像，做為主要的訓練資料，並使用了以下前處理及資料擴充方法: SpecAugment、Mixup、增加高斯雜訊 (Gaussian Noise) 、改變聲音的撥放速度 (Time Stretch) ，以及聲音平移 (Shift) ，將訓練資料擴充成四倍，並使用了 CNN 的 EfficientNet 預訓練模型以及 NoisyStudent 權重做為主要的模型架構，使用了交叉驗證，其中訓練資料與驗證資料為8:2，最後平均5個交叉驗證的模型輸出以及結果，作為模型的效能表現參考，其中Mean Val Accuracy = 0.912，Mean ROC Curve (AUC) = 0.985，在Private Leaderboard 中分數0.98113，排名27。

1. **資料處理**

資料的前處理主要透過Librosa完成，其中提取梅爾頻譜特徵的參數為sampling\_rate = 8000、n\_mels = 224、hop\_length = 179、n\_fft = 4480，接著提取梅爾頻譜的Delta Features以及Accelerate features (Delta\*Delta)，並將提取到的三張頻譜圖，標準化至0~255之間，並堆疊起來，便能得到224x224x3的三維影像作為訓練資料。資料擴充則是基於上述方法為基礎，使用了三種方法額外擴充了三倍資料，第一種為SpecAugment，將所提取到的三張頻譜圖，隨機遮蔽掉部分頻率以及時間軸，並將遮蔽過後的三張頻譜圖堆疊起來，作為新的訓練資料，能有效的降低模型過度擬合 (Over Fitting)的現象，並增加模型的穩定度，其中SpecAugment設定為隨機遮蔽頻率以及時間軸0~30%資料，重複以上動作兩次，也就是總共隨機遮蔽0~60%的資料，效果可參考下圖。

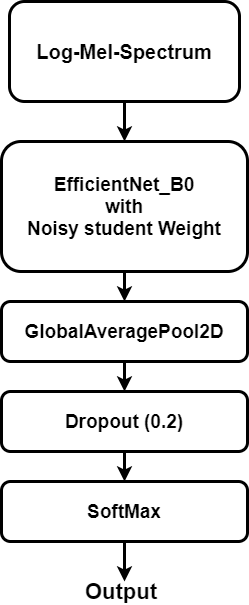
第二種方法為Mixup，是影像處理中常見的資料擴充方法，將兩張頻譜圖以隨機比例重新堆疊而成，標籤轉換成One Hot Encode後，根據比例重生成新的Soft Label。Mixup也能有效的降低模型過擬合，在訓練資料不足時能有效的擴充資料量，也能讓模型在判斷有多種環境聲音的音檔時有更全面的表現，效果可參考下圖。

第三種為增加高斯雜訊 (Gaussian Noise) 、改變聲音的撥放速度 (Time Stretch) ，以及聲音平移 (Shift)，其中高斯雜訊中amplitude = 0.015，改變聲音的撥放速度為1.25倍，聲音平移2.5秒，將每個音檔經過以上三種聲音處理後，做為額外擴充的訓練資料。經過以上三種擴充方法，可將資料擴充成四倍，如下圖。

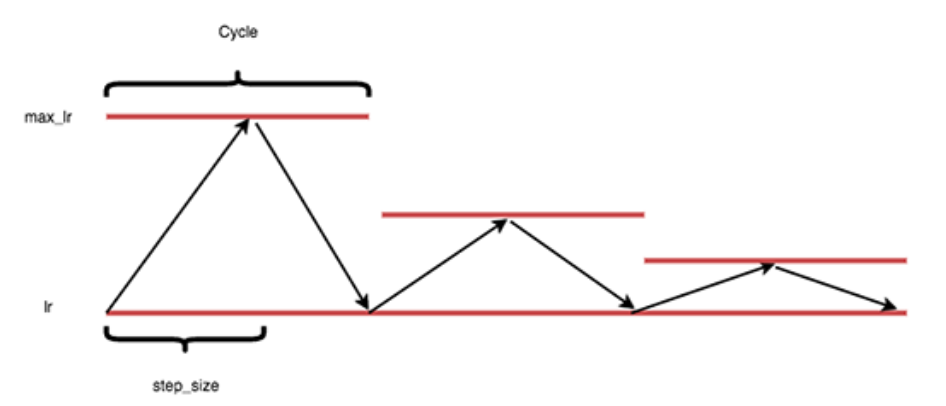


1. **模型架構**

模型架構主要為預訓練模型EfficientNet，其中分為B0~B6不同的模型深度，本隊使用B0作為主要架構，並使用了最新釋出的NoisyStudent模型權重，以此模型為基礎，增加GlobalAveragePooling2D、Dropout=0.2，最後使用Softmax輸出。



1. **訓練方式**

 本隊模型的Optimizer使用RAdam (Rectified Adam)，比起Adam能更快的收斂模型，達到較好的效果，learning rate使用Cyclical Learning Rate，其中參數max\_learning\_rate = 3E-5，min\_learning\_rate = 5E-5，Cycle次數為4次，epoch = 30，並在訓練完成時儲存準確率最高的Epoch作為模型的權重。傳統隨著Epoch逐漸下降的學習率，容易陷入局部最佳解 (Saddle point) ，且訓練時間較長，而Cyclical Learning Rate能根據Epoch改變不同的學習率，讓最佳的學習率更有機會套用到各個epoch中，並大幅減少訓練時間。學習率的變動可參考下圖。訓練模型使用交叉驗證，其中訓練資料與驗證資料為8:2，並平均5個交叉驗證的模型輸出以及結果，作為模型的效能表現參考。

1. **實驗分析與結論**

選擇EfficentNet的原因其一是此模型在ImageNet上擁有優異的表現

1. ，其二為EfficentNet提供了B0~B6不同深度的模型，未來在增加資料集的同時，也能同步提升模型深度，讓模型準確率更高，本隊使用B0做為主要的模型架構，交叉驗證後，平均五個模型結果為Mean Val Accuracy = 0.912，Mean ROC Curve (AUC) = 0.985。
2. **使用的外部資源與參考文獻**

外部資源:

MixupGenerator:

<https://github.com/yu4u/mixup-generator>

Audiomentations:

<https://github.com/iver56/audiomentations>

EfficientNet NoisyStudent Weight:

<https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet>

參考資料:

1. Hongyi Z., Moustapha C., Yann N. D. & David L. (2017) mixup: Beyond Empirical Risk Minimization.
2. Daniel S. P., William C., Yu Z., Chung C. C., Barret Z., Ekin D. C. & Quoc V. L. (2019) SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition.
3. Mingxing T. & Quoc V. L. (2019) EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks.
4. CNNs for Audio Classification

<https://towardsdatascience.com/cnns-for-audio-classification-6244954665ab>

1. MixUp augmentation for image classification

<https://keras.io/examples/vision/mixup/>

1. Image classification via fine-tuning with EfficientNet

<https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/>

1. mixup & cutout or random erasing to augment

<https://www.kaggle.com/daisukelab/mixup-cutout-or-random-erasing-to-augment>

1. Cyclical Learning Rates with Keras and Deep Learning

<https://www.pyimagesearch.com/2019/07/29/cyclical-learning-rates-with-keras-and-deep-learning/>

1. Super Convergence with Cyclical Learning Rates in TensorFlow

<https://towardsdatascience.com/super-convergence-with-cyclical-learning-rates-in-tensorflow-c1932b858252>

1. EfficientNetB5 with Keras (APTOS 2019)

<https://www.kaggle.com/carlolepelaars/efficientnetb5-with-keras-aptos-2019>

1. **心得感想與建議**

本次競賽本隊重心放在資料的處理及擴充上，但只用1200筆資料來預測3萬筆資料確實是有點吃力，或許應該把重心放在額外的資料集收集以及清洗上，能讓本次競賽的結果更好。另外Private LeaderBoard所提供的上傳時間有點短，且Private LeaderBoard開放上傳時所顯示的submission分數，與最終結算的分數落差較大，較難根據分數高低來選擇最終上傳的結果。