Configuration Manual

Evaluating performance of shuffling data augmentation techniques for audio event detection.
MSc Cloud Computing

David Kelly
Student ID: 13127390

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	David Kelly		
Student ID:	13127390		
Programme:	MSc Cloud Computing	**Year:**	2022
Module:	MSc Research Project		
Supervisor:	Vikas Sahni		
Submission Due Date:	10/11/2022		
Project Title:	Evaluating performance of mixing and shuffling data augmentation techniques for audio scene classification.		
Word Count:	1600		
Page Count	12		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: David Kelly

Date: 10/11/2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

David Kelly
13127390

A number of configuration steps are required to prepare a workstation to run the provided artefacts. This configuration manual assumes a Windows PC workstation with a CUDA capable Nvidia Graphics Card and a User Account with full UAC/Admin/Root access.

1 Workstation Specification

All development and testing activities were carried out on the following workstation.

Property	**Value**
Manufacturer	Dell
Model	XPS 7590
Form Factor	Laptop Workstation
CPU	Intel 9980HK, 2.4Ghz-5Ghz Clock
CPU Details	8 Core / 16 Thread
RAM	64 GB, DDR4 SODIMM
DISK	2TB NVME M.2 SSD
GPU	NVIDIA GTX 1650, 4GB GDDR5
OS	Windows 10 Professional, V22H2

2 Workstation Configuration

The experiment required a series of dependencies to be met on the underlying workstation. The figure below illustrates the base toolchain required for operation.

Property	**Value**
Nvidia GPU Driver	Version 516.59
Nvidia CUDA Toolkit	Version 9.2
Nvidia CUDNN Libraries	Version 7
Anaconda Package Manager	Version 2.2
Python	Version 3.6.13
Git	Latest

2.1 Graphics Driver and CUDA Acceleration

A base GPU driver was required to leverage GPU acceleration in the training stages. For the specific workstation GPU, the minimum compatible driver version of 396.26 was required to work in conjunction with the CUDA acceleration package version 9.2.
Installers are available online for common platforms[1][2].

2.2 CUDA Deep Neural Network Library (CUDNN)

In addition to the acceleration package CUDNN[3], a library of machine learning tools is required by Pytorch. In this instance, CUDNN 7 is compatible with CUDA 9.2 and driver version 396.26. The library needed to be installed manually; this means the contents of the /bin, /lib and /include CUDNN library folders were extracted and placed in the corresponding CUDA /bin /lib and /includes folders.

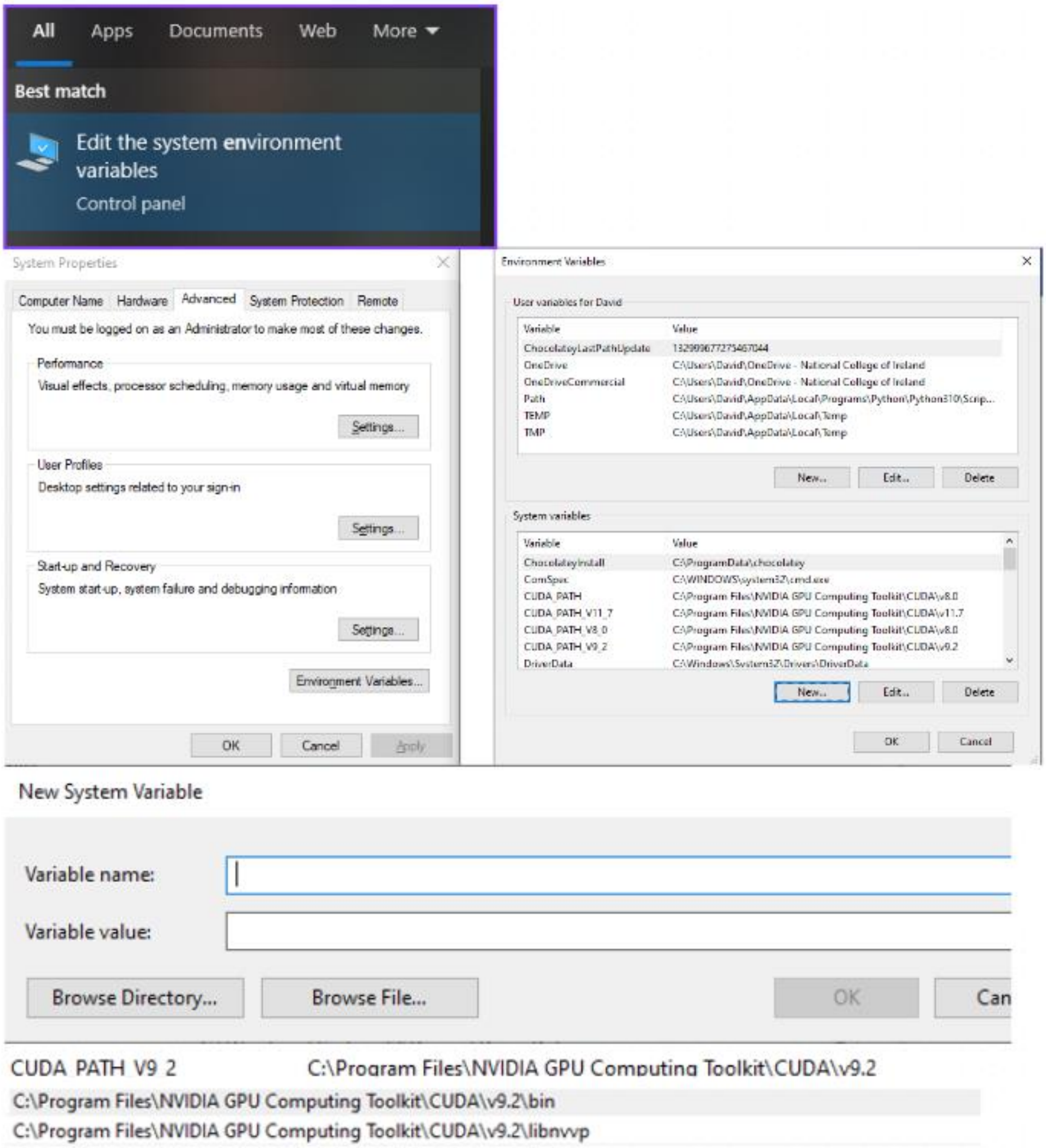
2.3 Environment Variables

The “System Path” and “Environment Variables” were required an update also. A pointer to the installed version of CUDA and its /bin locations. For Windows, this means the variable for CUDA_Path_V9_2 need to be set to the value of the folder where CUDA is installed. The system path required a reference to the CUDA /bin and /libnvp folders.

[1] https://www.nvidia.com/download/index.aspx

[2] https://developer.nvidia.com/cuda-downloads

[3] https://developer.nvidia.com/cudnn



System Properties

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage and virtual memory

User Profiles

Desktop settings related to your sign-in

Start-up and Recovery

System start-up, system failure and debugging information

Environment Variables

User variables for David

Variable	Value
ChocolateyLastPathUpdate	132999677275467044
OneDrive	C:\Users\David\OneDrive - National College of Ireland
OneDriveCommercial	C:\Users\David\OneDrive - National College of Ireland
Path	C:\Users\David\AppData\Local\Programs\Python\Python310\Scrip...
TEMP	C:\Users\David\AppData\Local\Temp
TMP	C:\Users\David\AppData\Local\Temp

System variables

Variable	Value
ChocolateyInstall	C:\ProgramData\chocolatey
ComSpec	C:\WINDOWS\system32\cmd.exe
CUDA_PATH	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0
CUDA_PATH_V11_7	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.7
CUDA_PATH_V8_0	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0
CUDA_PATH_V9_2	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.2
DriverData	C:\Windows\System32\Drivers\DriverData

New System Variable

Variable name:

Variable value:

CUDA_PATH_V9_2   C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.2

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.2\bin

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.2\libnvvp

Figure 1. Visual steps to add environment variables on Windows 10

2.4 VS Code

The IDE VS Code was the application host for the Jupyter Notebook artefact. For this experiment, it was essential VS Code was installed[4]. VS also requires the Jupyter extension to be installed. The extension is available from the Extension Marketplace sidebar in VS Code.

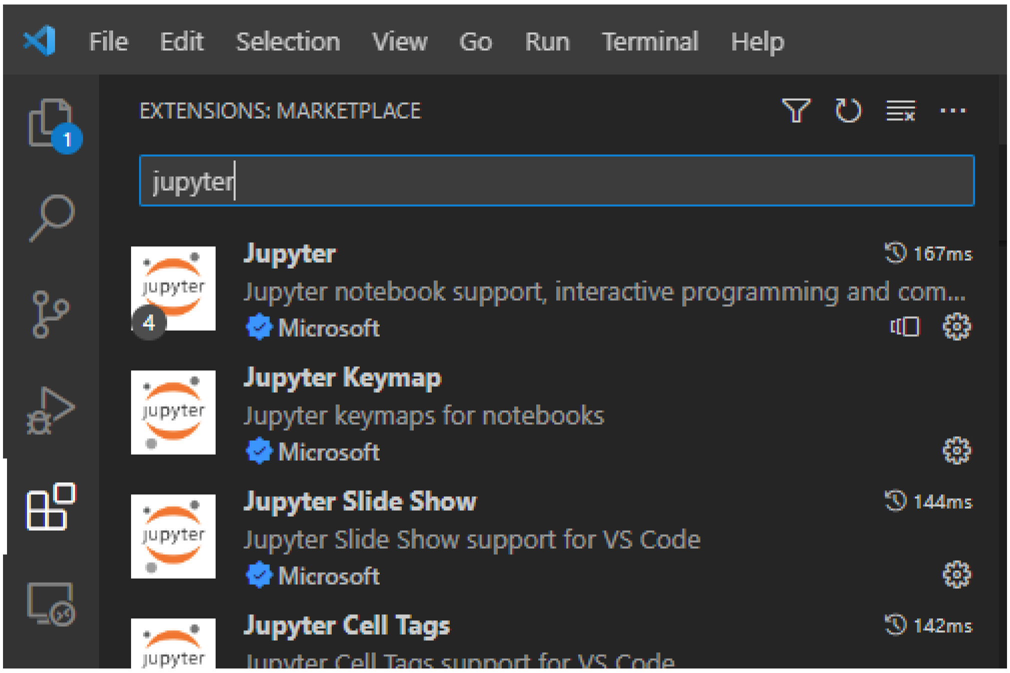


Figure 2. VS Code Extension Manager - Installing Jupyter

2.5 GIT

Git is required to download the latest version of the codebase. Git installers are available online[5].

2.6 PowerBI

PowerBI is required to analyse the resulting performance data. Installers for PowerBI are available for common platforms[6].

3 Platform

3.1 Clone Latest Code from GitHub Repository

From the terminal, the following command was executed to download the latest code:

[4] https://code.visualstudio.com/download
[5] https://git-scm.com/downloads
[6] https://powerbi.microsoft.com/en-us/downloads/

```
git clone https://github.com/davidlakelly/dcase2018_task5
```

3.2 Anaconda

Anaconda is a tool for streamlining package management in development settings. Download and run the Anaconda installer. Installers are available online for common platforms[7].

3.3 Anaconda Environment

The code repository contained a formatted yml file. Using the terminal, the working directory was changed into the repository. The following terminal command automatically configured python with the remaining necessary python packages to execute the augmentation and learning.

- `“conda env create -f environment.yml”`

Data Repository Setup

Two empty folders should exist with the code repository: dev and eval. Data from the development and evaluation SINS datasets should be extracted and placed into the respective empty folders.

3.3.1 Data Download

The SINS database is available online from https://zenodo.org/record/1247102. Both development and evaluation datasets were downloaded from here.

3.3.2 Data Extraction

The downloaded ZIP files should be located in the downloads folder. From this location, the development dataset was extracted to the empty dev folder code repository. This step was repeated for the evaluation dataset.

[7] https://www.anaconda.com/products/distribution

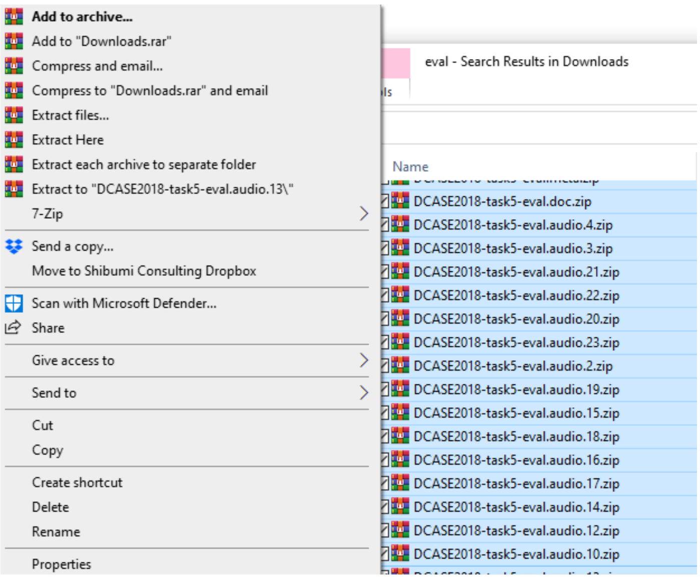

Add to archive...
Add to "Downloads.rar"
Compress and email...
Compress to "Downloads.rar" and email
Extract files...
Extract Here
Extract each archive to separate folder
Extract to "DCASE2018-task5-eval.audio.13\"
7-Zip
Send a copy...
Move to Shibumi Consulting Dropbox
Scan with Microsoft Defender...
Share
Give access to
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties
eval - Search Results in Downloads
Name
DCASE2018-task5-eval.doc.zip
DCASE2018-task5-eval.audio.4.zip
DCASE2018-task5-eval.audio.3.zip
DCASE2018-task5-eval.audio.21.zip
DCASE2018-task5-eval.audio.22.zip
DCASE2018-task5-eval.audio.20.zip
DCASE2018-task5-eval.audio.23.zip
DCASE2018-task5-eval.audio.2.zip
DCASE2018-task5-eval.audio.19.zip
DCASE2018-task5-eval.audio.15.zip
DCASE2018-task5-eval.audio.18.zip
DCASE2018-task5-eval.audio.16.zip
DCASE2018-task5-eval.audio.17.zip
DCASE2018-task5-eval.audio.14.zip
DCASE2018-task5-eval.audio.12.zip
DCASE2018-task5-eval.audio.10.zip

Figure 3. Select all relevant files and extract them to the dev and eval folders

3.3.3 Folder Structure Verification

The result of the extraction provides the following file system architecture on disk.

```
└───dcase2018_task5-master
    ├───.vscode
    ├───appendixes
    ├───dev
    │   └───DCASE2018-task5-dev
    │       ├───audio
    │       └───evaluation_setup
    ├───eval
    │   └───DCASE2018-task5-eval
    │       ├───audio
    │       └───evaluation_setup
    ├───features
    │   └───logmel
    ├───pytorch
    │   ├───main_pytorch
    │   │   ├───full_train
    │   │   └───holdout_fold1
    │   └───__pycache__
    └───utils
        └───__pycache__
```

To produce a similar diagram, enter the following terminal commands to change the directory to the code repository and use the below tree command to print the directory structure on disk.

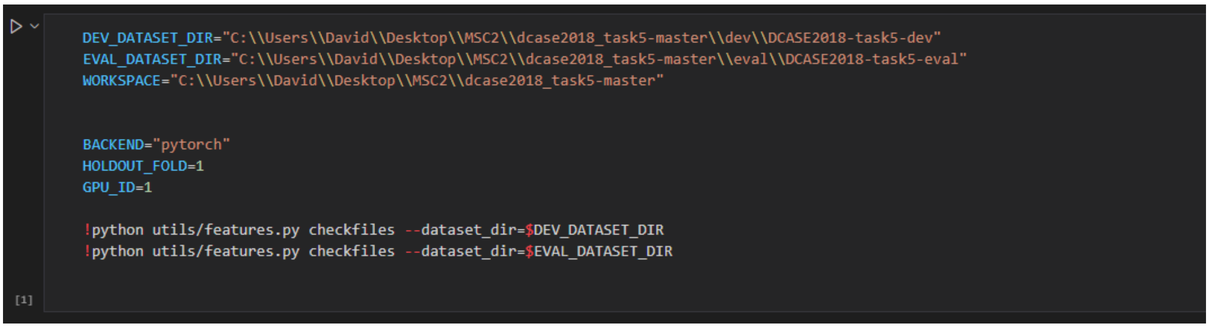
- `“cd dcase2018_task5-master”`
- `“tree"`

4 Experiment Execution

Several python scripts and functions are provided to facilitate the generation, training, and evaluation of datasets. This project contains a novel artefact in the form of a Jupyter notebook to data augmentation driver. Jupyter notebooks consist of cells; the contents of which is a combination of python or batch scripts. Opening the Jupyter Notebook in VS Code will allows the researcher to interact with the notebook and execute the cells.

4.1 Cell 1 – File Verification

- These cells executed a script inside the repository and scanned any available metadata files. The script also took the file names as specified in the metadata files and attempted to find those files on disk.



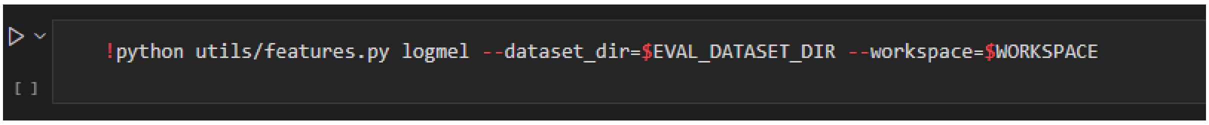
```
DEV_DATASET_DIR="C:\\Users\\David\\Desktop\\MSC2\\dcase2018_task5-master\\dev\\DCASE2018-task5-dev"
EVAL_DATASET_DIR="C:\\Users\\David\\Desktop\\MSC2\\dcase2018_task5-master\\eval\\DCASE2018-task5-eval"
WORKSPACE="C:\\Users\\David\\Desktop\\MSC2\\dcase2018_task5-master"


BACKEND="pytorch"
HOLDOUT_FOLD=1
GPU_ID=1

!python utils/features.py checkfiles --dataset_dir=$DEV_DATASET_DIR
!python utils/features.py checkfiles --dataset_dir=$EVAL_DATASET_DIR
```

4.2 Cell 2 – Extract Features for Eval Dataset

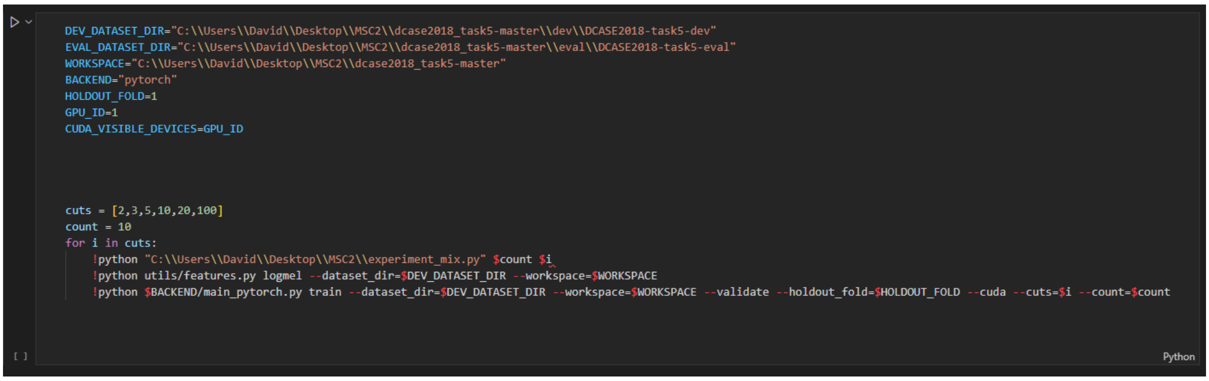
- Log Mel features were extracted from the available validation evaluation set.



```
!python utils/features.py logmel --dataset_dir=$EVAL_DATASET_DIR --workspace=$WORKSPACE
```

4.3 Cell 3 – Evaluating Performance (10% Additional Data)

- This cell executed a loop performing the following actions: it generated augmented data, extracted log-mel features and trained the network. The variables in the top of the cell should be noted at this point. These variables were injected into the scripts to be used as the augmentation parameters. In this cell, count=10 specified “every 10th file” to be chosen for augmentation from the underlying dataset.



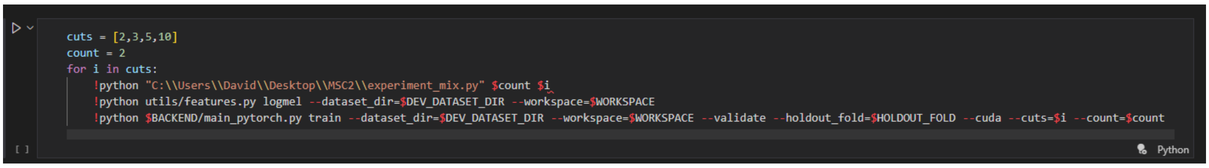
```
DEV_DATASET_DIR="C:\\Users\\David\\Desktop\\MSC2\\dcase2018_task5-master\\dev\\DCASE2018-task5-dev"
EVAL_DATASET_DIR="C:\\Users\\David\\Desktop\\MSC2\\dcase2018_task5-master\\eval\\DCASE2018-task5-eval"
WORKSPACE="C:\\Users\\David\\Desktop\\MSC2\\dcase2018_task5-master"
BACKEND="pytorch"
HOLDOUT_FOLD=1
GPU_ID=1
CUDA_VISIBLE_DEVICES=GPU_ID




cuts = [2,3,5,10,20,100]
count = 10
for i in cuts:
    !python "C:\\Users\\David\\Desktop\\MSC2\\experiment_mix.py" $count $i
    !python utils/features.py logmel --dataset_dir=$DEV_DATASET_DIR --workspace=$WORKSPACE
    !python $BACKEND/main_pytorch.py train --dataset_dir=$DEV_DATASET_DIR --workspace=$WORKSPACE --validate --holdout_fold=$HOLDOUT_FOLD --cuda --cuts=$i --count=$count
```

4.4 Cell 4&5 – Evaluating Performance (50% Additional Data)

- This cell contained a duplication of the initial evaluation cell but this time looking at scenarios where 50% data is added to the base dataset.



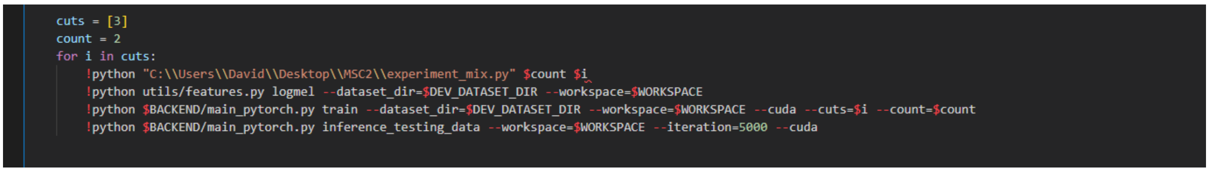
```
cuts = [2,3,5,10]
count = 2
for i in cuts:
    !python "C:\\Users\\David\\Desktop\\MSC2\\experiment_mix.py" $count $i
    !python utils/features.py logmel --dataset_dir=$DEV_DATASET_DIR --workspace=$WORKSPACE
    !python $BACKEND/main_pytorch.py train --dataset_dir=$DEV_DATASET_DIR --workspace=$WORKSPACE --validate --holdout_fold=$HOLDOUT_FOLD --cuda --cuts=$i --count=$count
```

4.5 Cell 6 – Evaluating Performance All Folds (N Additional Data)

- Throughout previous testing runs, performance would have been evaluated on one “holdout” fold. This was roughly 30% of the original dataset that was kept from the network training and instead used for validation scores. For reporting and comparison purposes, the best performing combination of augmentation and percentage of augmented data was evaluated in this cell on all 4 folds of holdout data.

4.6 Cell 7 – Evaluating Performance on Evaluation Dataset

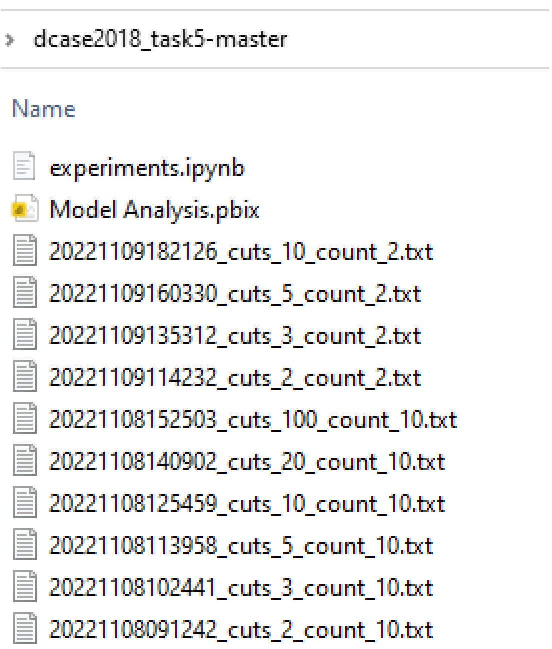
- The final cell trained the network on the full set of data from the SINS dataset, without holdout, along with the chosen best augmentation performer. The network was then evaluated on a dataset containing zero files the network has seen previously.



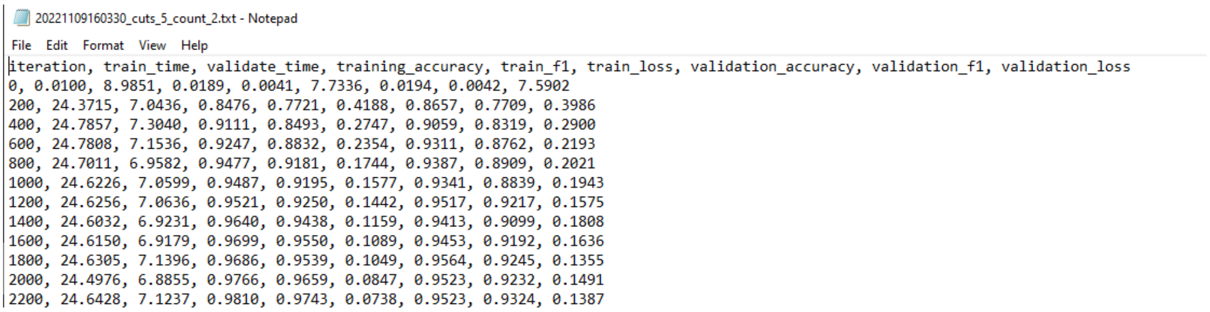
```
cuts = [3]
count = 2
for i in cuts:
    !python "C:\\Users\\David\\Desktop\\MSC2\\experiment_mix.py" $count $i
    !python utils/features.py logmel --dataset_dir=$DEV_DATASET_DIR --workspace=$WORKSPACE
    !python $BACKEND/main_pytorch.py train --dataset_dir=$DEV_DATASET_DIR --workspace=$WORKSPACE --cuda --cuts=$i --count=$count
    !python $BACKEND/main_pytorch.py inference_testing_data --workspace=$WORKSPACE --iteration=5000 --cuda
```

5 Experiment Results

The network was configured to log performance metrics during training. These metrics were stored in CSV files in the root of the project. Typical file name contain a timestamps and an indication of the chosen augmentation parameters for that result set. The contents of the files included a header row describing the metrics observed during training. Each subsequent row will contained the performance values for each of those variables at specified training intervals, roughly every 200 network iterations.



```
experiments.ipynb
Model Analysis.pbix
20221109182126_cuts_10_count_2.txt
20221109160330_cuts_5_count_2.txt
20221109135312_cuts_3_count_2.txt
20221109114232_cuts_2_count_2.txt
20221108152503_cuts_100_count_10.txt
20221108140902_cuts_20_count_10.txt
20221108125459_cuts_10_count_10.txt
20221108113958_cuts_5_count_10.txt
20221108102441_cuts_3_count_10.txt
20221108091242_cuts_2_count_10.txt
```



20221109160330_cuts_5_count_2.txt - Notepad
File Edit Format View Help

```
iteration, train_time, validate_time, training_accuracy, train_f1, train_loss, validation_accuracy, validation_f1, validation_loss
0, 0.0100, 8.9851, 0.0189, 0.0041, 7.7336, 0.0194, 0.0042, 7.5902
200, 24.3715, 7.0436, 0.8476, 0.7721, 0.4188, 0.8657, 0.7709, 0.3986
400, 24.7857, 7.3040, 0.9111, 0.8493, 0.2747, 0.9059, 0.8319, 0.2900
600, 24.7808, 7.1536, 0.9247, 0.8832, 0.2354, 0.9311, 0.8762, 0.2193
800, 24.7011, 6.9582, 0.9477, 0.9181, 0.1744, 0.9387, 0.8909, 0.2021
1000, 24.6226, 7.0599, 0.9487, 0.9195, 0.1577, 0.9341, 0.8839, 0.1943
1200, 24.6256, 7.0636, 0.9521, 0.9250, 0.1442, 0.9517, 0.9217, 0.1575
1400, 24.6032, 6.9231, 0.9640, 0.9438, 0.1159, 0.9413, 0.9099, 0.1808
1600, 24.6150, 6.9179, 0.9699, 0.9550, 0.1089, 0.9453, 0.9192, 0.1636
1800, 24.6305, 7.1396, 0.9686, 0.9539, 0.1049, 0.9564, 0.9245, 0.1355
2000, 24.4976, 6.8855, 0.9766, 0.9659, 0.0847, 0.9523, 0.9232, 0.1491
2200, 24.6428, 7.1237, 0.9810, 0.9743, 0.0738, 0.9523, 0.9324, 0.1387
```

6 PowerBI Model

To facilitate easy comparison between the experiment results, a PowerBI model is provided with preconfigured dashboards to analyse the experimental results. The following steps demonstrate how to configure the model to accept additional experimental results files.

6.1 Data Loading

- For each file you wish to analyse, select “Get Data” from the Ribbon of PowerBI.
- Choose Text/CSV.
- Select the desired file.

6.2 Relationship Mapping

- The key relationship between each experiment is the iteration variable. From the relationship tab choose “Manage Relationships” from the Ribbon. From this window you can specify a relationship between two data tables.
- Select “New Relationship”.
- Choose the baseline dataset as the origin for the relationship.

- Choose a different dataset as the destination for the relationship.
- Set cardinality to “1:1” and Cross Filter to “Both”
- Repeat this step for each file you wish to analyse.