CSC1028 AutoML

Test Plan

kaylem mcshane 40288049
kmcshane10@qub.ac.uk

CSC1028 CS Challenges Test Plan

Brief

Developing machine learning models to solve problems is often described as more of an art than a science. This is another way of saying that it is very hard to guess how well a given technique will work on a problem. AI developers therefore spend a lot of their time searching over different machine learning architectures and hyper parameters trying to find ones that work well for their problem. This can lead to messy code and create a feeling like you are building on sand, as soon as you decide that one part of the model is working well changing another part may invalidate that choice. This has led to the field of AutoML which is a process of automatically searching for good AI architecture and hyper parameters. This approach is a good way to approach AI problems where instead of writing a single algorithm that you change you develop a python script that can generate different AI experiments with different architectures and parameters. Using this approach you can automatically run lots of different experiments and automatically search for the best approach. As you work on the project you are adding different alternative architectures to your python script so that you can automatically search over more approaches. In this way you never take a step back, every choice you add is at worst making your system try unhelpful approaches but the optimal design can be easily produced from your configuration.

The goal of this project is to take one or more of the AI algorithms described as part of the FastAI course (the leading course on Deep learning which includes a number of state of the art models for solving common machine learning problems). For each problem express the choices available in how the algorithm is implemented using a csv format (like a row in excel).

Make the resulting training algorithms capable of running on google colab and as a docker gpu container so that the machine learning algorithms can be easily trained on a cloud server.

Mulitple students can work on this project each focused on a different AI problem being configured e.g. Image recognition, time series prediction, object detection etc.

CSC1028 CS Challenges Test Plan

Requirements

python 3.8*

pip 20.3.4*

Cuda 10.1*

fastai 2.2.7

fastaudio 0.1.4

torch 1.8.0

torchvision 0.9.0

torchaudio 0.8.0

librosa 0.8.0

*On the initial start up all required dependencies will be installed within a python virtual environment, the only key requirements before beginning the testing is to have the noted versions of python, pip and Cuda.

Cuda download: https://developer.nvidia.com/cuda-10.1-download-archive-base

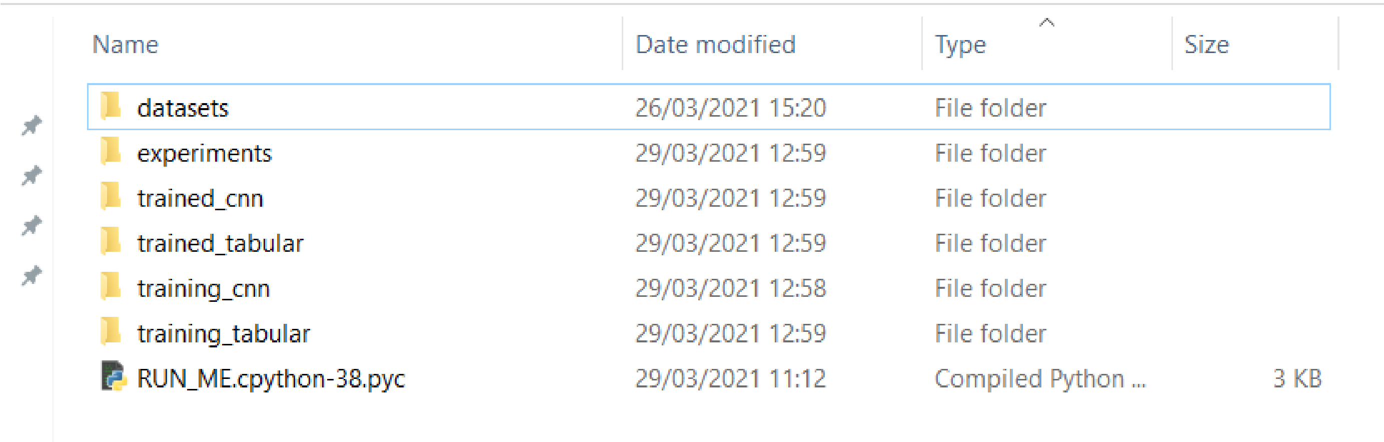
CSC1028 CS Challenges Test Plan

Getting Started

Download the project zip file from this link:

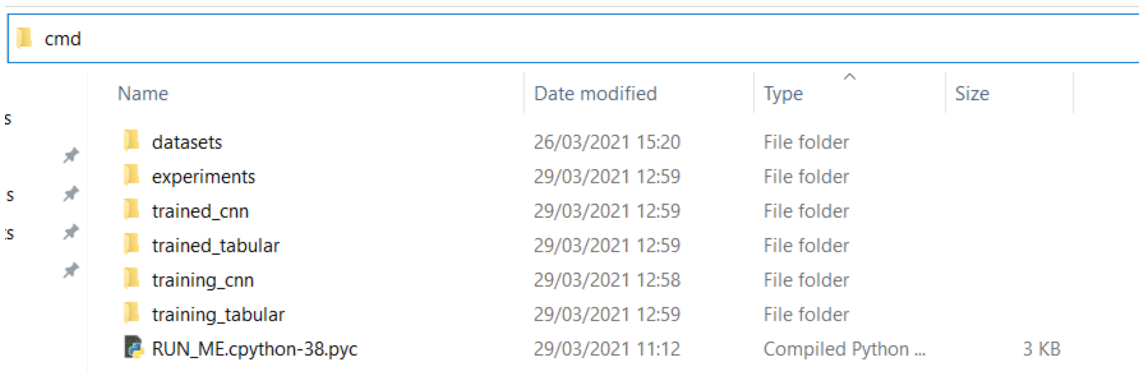
Unzip the project in a standard location, it is recommended to do this within the C drive but this is not essential as the code is designed to run within any directory.

The initial directory should be viewed as this:



Note: for ease of viewing error messages, for the purpose of this test it is recommended that the initial python executable be called within the command line. The code can be called by double clicking the RUN_ME file but this may lead to the CLI closing before the error can be viewed.

The command line can be opened by clicking the file path and typing ‘cmd’.



The file can then by executed by typing ‘python RUN_ME.cpython-38.pyc’

CSC1028 CS Challenges Test Plan

Test Case 1: VENV Generation

When the code is called for the first time it will check for the existence of the python virtual environment. If this is not yet available it will generate one for the user, installing all necessary dependencies and relevant versions.

User Action:

'python RUN_ME.cpython-38.pyc (within CLI)'

Expected output:

All dependencies installed, VENV present in directory

Name	Date modified	Type	Size
auto-ml-env	29/03/2021 12:31	File folder	
datasets	29/03/2021 14:08	File folder	
experiments	29/03/2021 12:59	File folder	
trained_cnn	29/03/2021 12:59	File folder	
trained_tabular	29/03/2021 12:59	File folder	
training_cnn	29/03/2021 12:58	File folder	
training_tabular	29/03/2021 12:59	File folder	
RUN_ME.cpython-38.pyc	29/03/2021 11:12	Compiled Python ...	3 KB

Test Case 2: Running program with existing VENV

Too ensure the program is not downloading files unnecessarily, the VENV is only generated when it is required.

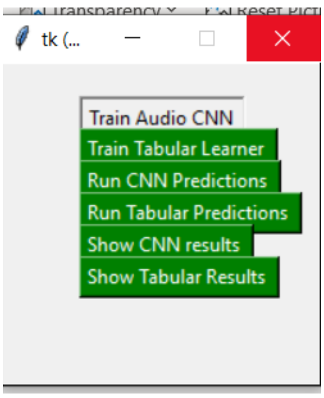
User Action:

Close programme

Execute *'python RUN_ME.cpython-38.pyc (within CLI)'*

Expected Output:

The GUI should open with no new dependency downloads.



CSC1028 CS Challenges Test Plan

Test Case 3: Training a CNN model on preloaded data

When the project is downloaded, there will be a predownloaded dataset available on Kaggle: https://www.kaggle.com/kinguistics/heartbeat-sounds

It contains data in the necessary format and should run instantly using the wav files and csv files stored within datasets→train.

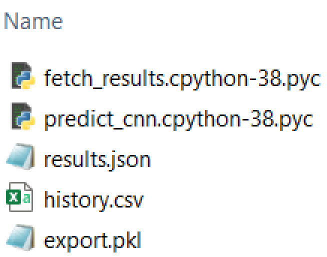
User Action:

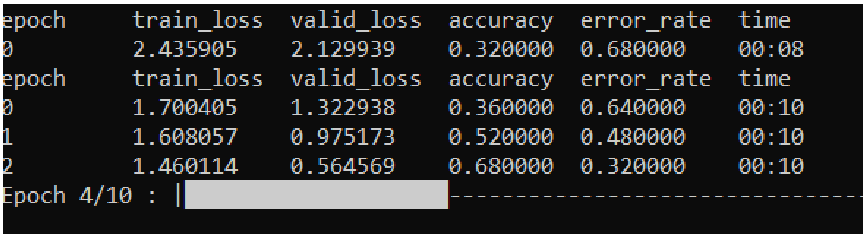
Press ‘Train Audio CNN’ Button

Expected Outcome:

FastAi’s model training function should be printed and displayed,

Within the trained_cnn folder, export.pkl file and results JSON should be produced.





Test Case 4: Training a Tabular Model on preloaded data

Uses the same sample as the last test case to train a tabular model.

User Action:

Press ‘Train Tabular Learner’ Button

Expected Outcome:

CSC1028 CS Challenges Test Plan

FastAi's model training function should be printed and displayed,

within the trained_tabular folder, export.pkl file and results JSON should be produced.

Name

- export.pkl
- fetch_results.cpython-38.pyc
- history.csv
- results.json

```
epoch     train_loss  valid_loss  accuracy  time
0         1.407914    1.398030    0.061224  00:00
epoch     train_loss  valid_loss  accuracy  time
0         1.329216    1.396086    0.061224  00:00
1         1.241488    1.394207    0.061224  00:00
2         1.172004    1.392155    0.061224  00:00
3         1.099365    1.390169    0.061224  00:00
4         1.030550    1.388189    0.061224  00:00
5         0.971375    1.386284    0.265306  00:00
6         0.917265    1.384455    0.265306  00:00
```

Test Case 5: Run CNN Predictions on a directory

The program can run predictions on audio files contained within a fixed directory based on the user input for the source and output directories.

User Action:

Press 'Run CNN Predicitions'

Enter c:\....\autoMl\datasets\test

Enter c:\....\autoMl\datasets\test

Expected Outcome:

Predictions are printed in the CL

JSON files are generated in the output location

Name	#	Ti
artifact__201012172...		
artifact__201012172...		
artifact__201105040...		
artifact__201105040...		
artifact__201105041...		
artifact__201105041...		
artifact__201105051...		
artifact__201105051...		
artifact__201105060...		
artifact__201105060...		

```
Enter the address of the file to run predictions on:C:\autoMl\datasets\test
Enter the address to output results too:C:\autoMl\datasets\test
export.pkl
fetch_results.cpython-38.pyc
history.csv
predict_cnn.cpython-38.pyc
results.json
artifact__201012172012.wav Prediction: artifact; Probability: 0.8852
artifact__201105040918.wav Prediction: artifact; Probability: 0.9998
artifact__201105041959.wav Prediction: artifact; Probability: 1.0000
artifact__201105051017.wav Prediction: artifact; Probability: 0.9999
artifact__201105060108.wav Prediction: artifact; Probability: 0.9991
```

CSC1028 CS Challenges Test Plan

Test Case 6: Print results from a training model

After training a model, the results are stored within a JSON file and can be retrieved using the corresponding buttons.

User Action:

Press 'Show CNN Results'

'Show Tabular Results'

Expected Output:

Prints out the corresponding results

	epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0	1.700405	1.322938	0.36	0.64	00:10
1	1	1.608057	0.975173	0.52	0.48	00:10
2	2	1.460114	0.564569	0.68	0.32	00:10
3	3	1.233939	0.510785	0.88	0.12	00:10
4	4	1.110408	0.359951	0.88	0.12	00:10
5	5	1.013931	0.505373	0.72	0.28	00:11
6	6	0.919917	0.472999	0.84	0.16	00:10
7	7	0.845358	0.451832	0.76	0.24	00:10
8	8	0.802791	0.652589	0.80	0.20	00:10
9	9	0.778321	0.557619	0.76	0.24	00:11

	epoch	train_loss	valid_loss	accuracy	time
0	0	1.329216	1.396086	0.061224	00:00
1	1	1.241488	1.394207	0.061224	00:00
2	2	1.172004	1.392155	0.061224	00:00
3	3	1.099365	1.390169	0.061224	00:00
4	4	1.030550	1.388189	0.061224	00:00
5	5	0.971375	1.386284	0.265306	00:00
6	6	0.917265	1.384455	0.265306	00:00
7	7	0.866477	1.382631	0.265306	00:00
8	8	0.819619	1.380697	0.265306	00:00
9	9	0.777894	1.378694	0.265306	00:00

Test Case 7: Run CNN Learner with no dataset

Ensure the project runs and is sustainable when no dataset is present

User Action:

Delete all files from datasets→train

Press 'Train Audio CNN'

Expected Outcome:

Error stack trace

Program does not crash

CSC1028 CS Challenges Test Plan

Test Case 8: Run Tabular Learner with no dataset

Ensure the project runs and is sustainable when no dataset is present

User Action:

Delete all files from datasets→train

Press ‘Train Tabular Model’

Expected Outcome:

Error stack trace

Program does not crash

Test Case 9: Run CNN Prediction with no data

Ensure the project runs and is sustainable when no dataset is present

User Action:

Delete all files from datasets→test

Press ‘Run CNN Predictions’

C:\autoMl\datasets\test

C:\autoMl\datasets\test

Expected Outcome:

Error stack trace

Program does not crash

CSC1028 CS Challenges Test Plan

Test Case 10: Run CNN Prediction with null paths

Ensure the project runs and is sustainable when no dataset is present

User Action:

Press ‘Run CNN Predictions’

“”

“”

Expected Outcome:

Prompts user to enter non-null value

Program does not crash

Test Case 11: Fetch results when results.json is unavailable

User Action:

Remove results.json from trained_cnn

Remove results.json from trained_tabular

Press ‘Show CNN Results’

Press ‘Show Tabular Results’

Expected Outcome:

Error stack trace

Program does not crash

The next test cases will assess the systems ability to manage unknown datasets which can be selected by the tester, it is recommended to use Kaggle to download the dataset due to the range of available datasets. The requirements for the datasets are as follows:

- The dataset must consist of at least 128 wav files and 1 csv file
- The csv file MUST contain a column named ‘filename’ and a column called ‘label’ which store the corresponding filenames and labels for the

dataset. Whilst this might not be available from the source, the dataset can be manually adjusted within the csv file as needed.

- All files must be deleted from datasets→train.
- The selected dataset must be dumped within datasets→ train
- The dataset must contain no sub-directories, all files must be stored within ..\datasets\train

Datasets can be found here: https://www.kaggle.com/search?q=audio+data+tag%3A%22audio+data%22

Test Case 12: Training a CNN model on new data

When the project is downloaded, there will be a predownloaded dataset available on Kaggle: https://www.kaggle.com/kinguistics/heartbeat-sounds

It contains data in the necessary format and should run instantly using the wav files and csv files stored within datasets→train.

User Action:

Press ‘Train Audio CNN’ Button

Expected Outcome:

FastAi’s model training function should be printed and displayed,

Within the trained_cnn folder, export.pkl file and results JSON should be produced.

Test Case 13: Training a Tabular Model on new data

Uses the same sample as the last test case to train a tabular model.

User Action:

Press ‘Train Tabular Learner’ Button

Expected Outcome:

FastAi’s model training function should be printed and displayed, within the trained_tabular folder, export.pkl file and results JSON should be produced.

Test Case 14: Run CNN Predictions on new data

The program can run predictions on audio files contained within a fixed directory based on the user input for the source and output directories.

User Action:

Dump some files in datasets→test

Press ‘Run CNN Predicitions’

Enter c:\....\autoMl\datasets\test

Enter c:\....\autoMl\datasets\test

Expected Outcome:

Predictions are printed in the CL

JSON files are generated in the output location