

Technical documentation

Contents

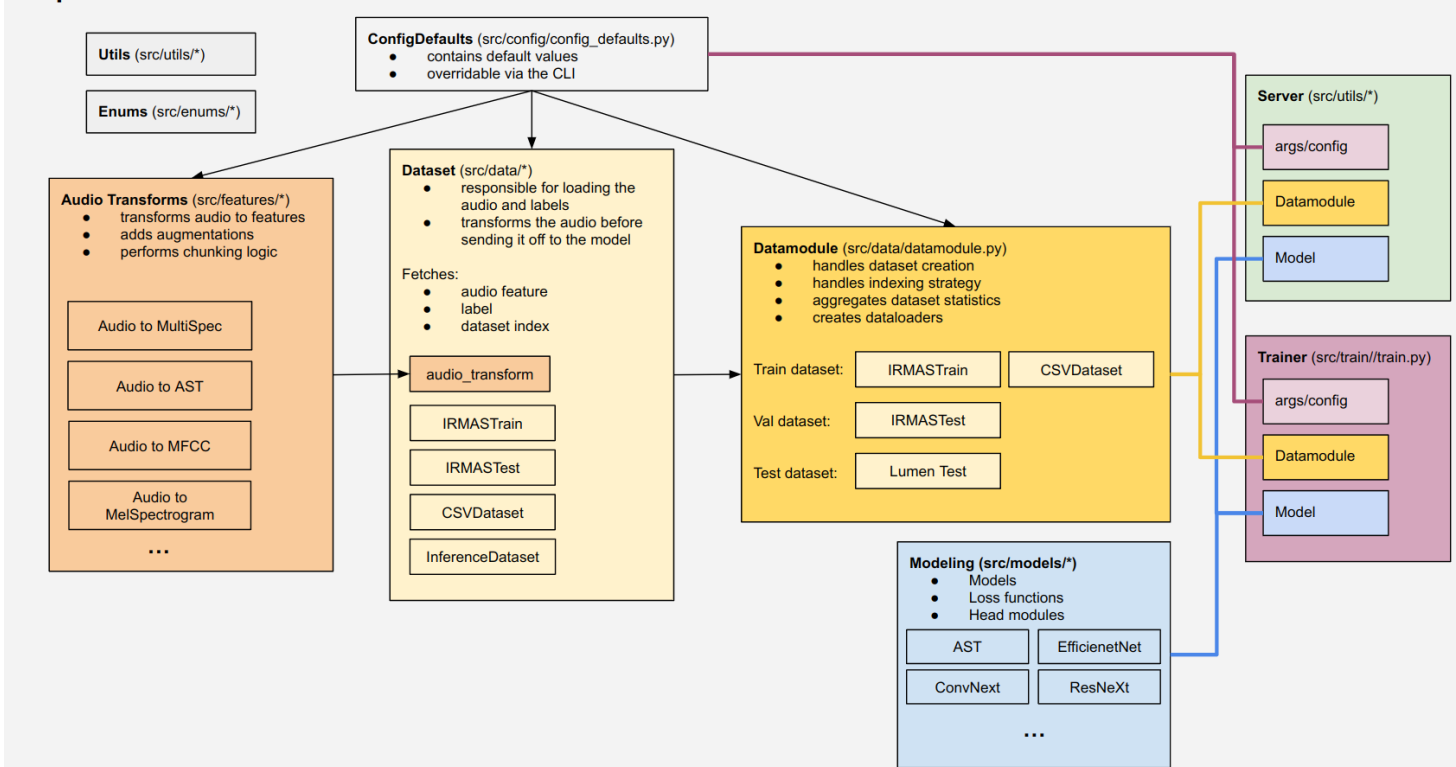
1 Source code architecture	2
2 External dependencies	2
3 Virtual Environment	2
3.1 a) Venv	2
3.2 b) Conda Virtual Environment	2
4 Dataset setup	3
4.1 Dataset directory types	3
4.2 Dataset directory type: <code>inference</code>	3
4.3 Dataset directory type: <code>irmastrain</code>	3
4.4 Dataset directory type: <code>irmastest</code>	4
4.5 Dataset directory type: <code>csv</code>	4
5 Pretrained models	4
6 Testing and Inference	5
6.1 a) Testing or Inference via REST API	5
6.2 b) Inference via the Python script (<code>src/inference/run_test.py</code>)	8
7 Training	10
7.1 Training logs	11
8 Tensorboard logs	11

Directory	Description
<code>data</code>	datasets and other data
<code>docs</code>	documentation
<code>figures</code>	figures
<code>models</code>	model checkpoints, model metadata
<code>src</code>	python source code
<code>src/server</code>	python server source code

Note: Your current directory in CLI should be the root of this project directory.

1 Source code architecture

Simplified code architecture



2 External dependencies

External dependencies we used (note that you don't need them for training and inference):

- FFmpeg: <https://ffmpeg.org/download.html>

3 Virtual Environment

Create and populate the **virtual environment**. Simply put, the virtual environment allows you to install Python packages for this project only (which you can easily delete later). This way, we won't clutter your global Python packages.

Choose one of two approaches to create the virtual environment:

3.1 a) Venv

help: <https://docs.python.org/3/library/venv.html>

Step 1: Create virtual env, activate it and install the packages:

```
python3 -m venv venv
source venv/bin/activate
sleep 1
pip install -r requirements.txt
```

Step 2: Install current directory as a editable Python module:

```
pip install -e .
```

3.2 b) Conda Virtual Environment

help: <https://conda.io/projects/conda/en/latest/user-guide/g#etting-started.html>

Step 1: Create virtual env, activate it and install the packages:

```
conda create --name lumen-env python==3.10
conda activate lumen-env
sleep 1
pip install -r requirements.txt
```

Step 2: Install current directory as a editable Python module:

```
pip install -e .
```

4 Dataset setup

This project allows training and inference on multiple different datasets. Although you can specify the dataset to any directory, we suggest moving your datasets to `data` directory.

Datasets are almost always specified in the following format when using the CLI:

```
--train-paths <DATASET_DIR_TYPE_1>:<DATASET_PATH_1> <DATASET_DIR_TYPE_2>:<DATASET_PATH_2>
e.g
--train-paths irmastrain:data/irmas/train csv:data/hello.csv
```

4.1 Dataset directory types

Each dataset directory type is a strategy for loading labels. `inference` dataset directory type doesn't load labels and it should be used for inference. Other dataset directory types (`irmastrain` , `irmatest` or `csv`) have built-in strategy which assign a label to audio. This is usually done by extracting the label from filename, .txt file or .csv columns.

4.2 Dataset directory type: `inference`

`inference` is restricted only for inference. Use this dataset directory type if you want to infer labels without calculating metrics.

It accepts a directory where we recursively find all audio files:

```
├── dataset
│   ├── file_1.wav
│   ├── file_2.wav
│   └── subdir
│       ├── file_3.ogg
│       ├── file_4.ogg
│       └── file_5.mp3
└── file_6.wav
```

`inference` also accepts a .csv file with the following structure:

file	ignored_col_1	ignored_col_2
dataset/483840.wav	"X"	"X"
dataset/3840.wav	"X"	"X"
dataset/145920.wav	"X"	"X"
dataset/126720.wav	"X"	"X"
dataset/195840.wav	"X"	"X"
dataset/34560.wav	"X"	"X"
dataset/226560.wav	"X"	"X"

4.3 Dataset directory type: `irmastrain`

This dataset directory type is used when you want to train IRMAS train-like directory. Subdirectories denote the instrument class. Wav filename format:

- `<PREFIX>[<INSTRUMENT>][<DRUMS>][<GENRE><suffix>.wav.`
- `<PREFIX>[<INSTRUMENT>][<GENRE><suffix>.wav.`

Directory structure:

```

├── dataset
│   ├── cel
│   │   ├── 008_[cel][nod][cla]0058__1.wav
│   │   ├── 008_[cel][nod][cla]0058__2.wav
│   │   ├── 008_[cel][nod][cla]0058__3.wav
│   │   ├── 012_[cel][nod][cla]0043__1.wav
│   │   └── ...
│   ├── cla
│   └── ...
└── voi

```

4.4 Dataset directory type: `irmastest`

IRMAS test-like directory Wav/label filename format:

- `<FILENAME>-<SONG_PART>.{wav,txt}`

Directory structure:

```

├── dataset
│   ├── 0050_10CC__I_M_NOT_IN_LOVE-11.txt
│   ├── 0050_10CC__I_M_NOT_IN_LOVE-11.wav
│   ├── 0050_10CC__I_M_NOT_IN_LOVE-13.txt
│   ├── 0050_10CC__I_M_NOT_IN_LOVE-13.wav
│   ├── 0050_10CC__I_M_NOT_IN_LOVE-15.txt
│   ├── 0050_10CC__I_M_NOT_IN_LOVE-15.wav
│   └── 0050_10CC__I_M_NOT_IN_LOVE-17.txt

```

Label file format:

```

cel
gel
voi

```

4.5 Dataset directory type: `CSV`

CSV File structure: `file` column contains audio filenames while other 11 columns denote the presence of the instrument in the audio file.

file	cel	cla	flu	gac	gel	org	pia	sax	tru	vio	voi
dataset/483840.wav	1	0	0	0	0	1	0	0	0	0	1
dataset/3840.wav	0	1	0	0	0	0	0	0	0	0	1
dataset/145920.wav	0	1	0	0	0	0	0	0	0	0	1
dataset/126720.wav	0	0	0	0	0	0	1	0	0	0	0
dataset/195840.wav	0	0	1	0	0	0	1	0	0	0	0
dataset/34560.wav	0	0	0	0	1	0	0	1	0	0	0
dataset/226560.wav	0	1	0	0	0	0	0	0	0	0	0

5 Pretrained models

Although you can specify model directory path with `--path-models` we suggest moving your pretrained models to the `models/` directory. Pretrained models should end with `.ckpt` extension which differentiates them from other files in `models/` directory, such as .txt log files. Example of the usual model filename `models/04-20-15-52-55_experiment/checkpoints/04-20-15-52-55_experiment.ckpt`

Before moving on, please ensure that you have at least one `.ckpt` pretrained model in your `models` directory. `.ckpt` file can be in a nested directory.

6 Testing and Inference

Testing: predict labels for a given model checkpoint and a given dataset and **calculate metrics** by comparing true label and predicted label

Inference: predict labels for a given model checkpoint and a given dataset

6.1 a) Testing or Inference via REST API

Help text for `src/server/main.py` :

```
usage: main.py [--host str] [--hot-reload] [--device str] --model-dir Path [--batch-size int]
[--port int] [--num-workers int] [-h] [--add-instrument-metrics]

options:
-h, --help
    show this help message and exit

ConfigDefault ['Config arguments']:
--batch-size int
    (default: 4)
--num-workers int
    Number of workers (default: 4)

Script arguments:
--host str
    Server host. (default: localhost)
--device str
    The device to be used eg. cuda:0. (default: cuda:0)
--hot-reload
    Use hot reload which auto-restarts the server on code change(not recommended).
    (default: False)
--model-dir Path
    Directory which to model checkpoints (.ckpt files) (default: None)
--port int
    Server port. (default: 8090)
--add-instrument-metrics
    Add instrument metrics to the response. This will increase number
    of items in API response size by (instrument count * number of metrics) (default: False)
```

Run the server:

```
python3 src/server/main.py --model-dir models/ --host localhost \
--port 8090 --batch-size 2 --num-workers 4 --add-instrument-metrics
```

Open <http://localhost:8090/docs> in browser. Click `GET /models` and “Try it out” button

Fast and Fourier audio prediction API 0.1.0 QAS3

openapi.json

API used for testing and predicting audio data on trained models.

```
usage: python3 src/server/main.py [--hot-reload] [--port int] [--host str] --model-dir Path [--device str] [--add-instrument-metrics] [-h]

options:
  -h, --help            show this help message and exit

Script arguments:
  --hot-reload            Use hot reload which auto-restarts the server on code change(not recommended). (default: False)
  --host str             Server host. (default: localhost)
  --model-dir Path       Directory which to model checkpoints (.ckpt files) (default: None)
  --device str           The device to be used eg. cuda:0. (default: cuda:0)
  --add-instrument-metrics Add instrument metrics to the response. This will increase number of items in API response size by (instrument count * number of metrics)
  (default: False)
  --port int            Server port. (default: 8090)
```

Quick instructions:

1. Make sure you have a `.ckpt` model inside of the `--model-dir` directory.
2. Click on `GET /models`.
3. Send a request to get all available models.
4. Copy the model path you want to use for prediction.
5. Click on `POST /model/predict-files`.
6. Paste the model path into the `model_checkpoint` field.
7. Choose the audio files you want to predict.
8. Predict labels for the audio files.

Resources

Models and datasets available for inference.

GET /models Get Models

Returns paths to all available models on the server. Use these model paths for all `POST` request predictions. Models are fetched from the directory `--model-dir`. Only models with the extension `.ckpt` are fetched.

Parameters

No parameters

Try it out

Click “Execute”.

Resources

Models and datasets available for inference.

GET /models Get Models

Returns paths to all available models on the server. Use these model paths for all `POST` request predictions. Models are fetched from the directory `--model-dir`. Only models with the extension `*.ckpt` are fetched.

Parameters

No parameters

Cancel

Execute

Responses

Code	Description	Links
200	Successful Response	No links

Media type

application/json

Controls Accept header.

Example Value

Schema

```
{
  "string"
}
```

Copy paste the path of the model you want to use for testing or prediction.

Server response

Code	Details
200	<p>Response body</p> <pre>["models/efnet2best/05-06-19-08-32_SlickDelta_efficient_net_v2_s/checkpoints/05-06-19-08-32_SlickDelta_efficient_net_v2_s_val_acc_0.3702_val_loss_0.7923.ckpt", "models/05-08-16-06-00_SlickDelta_convnext_small/checkpoints/05-08-16-06-00_SlickDelta_convnext_small_val_acc_0.0667_val_loss_0.3466.ckpt", ...]</pre> <p>Response headers</p> <pre>content-length: 4544 content-type: application/json date: Mon, 08 May 2023 19:10:44 GMT server: uvicorn</pre>

Go to the **POST /model/predict-directory** section, fill out the request parameters and predict the dataset for a given model.

POST
/model/predict-directory
Predict Directory

Infers classes of audios from a directory.You don't have to specify the dataset structure type is always **inference**

Parameters
Cancel
Reset

Name	Description
model_ckpt_path string(\$path) (query)	<div>models/efnet2best/05-06-19-08-32_SlickDelt</div> <div>Paste model path checkpoint</div>

Request body
application/json

["data/ixmas/test"]

Change path of a directory or a .csv file

Execute

Clear

Check the response body.

Server response

Code

Details

200

Response body

```
{
  "103 - love spirals downwards - will you fade-12": {
    "cel": 1,
    "cla": 1,
    "flu": 1,
    "gac": 1,
    "gel": 1,
    "org": 1,
    "pia": 1,
    "sax": 1,
    "tru": 1,
    "vio": 1,
    "voi": 0
  },
  "08 - Mum - Weeping Rock Rock-25": {
    "cel": 1,
    "cla": 0,
    "flu": 1,
    "gac": 1,
    "gel": 0,
    "org": 1,
    "pia": 1,
    "sax": 0,
    "tru": 0,
  }
```

For testing it's use `POST /model/test-directory` and fill out the `dataset_type` parameter in the body with some of the following: [`irmastrain` , `irmastest` , `csv`]

6.2 b) Inference via the Python script (`src/inference/run_test.py`)

It's also possible to use a python script to test and

```
usage: python3 src/inference/run_test.py [--batch-size int] [--ckpt [Path]] [--device str]
[--save-roc] [--out-dir Path] [--num-workers int] [-h] [--save-metric-hist] [--save-confusion]
[--dataset-paths [List]]
```

options:

-h, --help show this help message and exit

ConfigDefault ['Config arguments']:

--batch-size int

(default: 4)

--ckpt [Path]

.ckpt file, automatically restores model, epoch, step, LR schedulers, etc... (default: None)

--num-workers int Number of workers (default: 4)

--dataset-paths [List]

Dataset path with the following format format:

--dataset-paths inference:/path/to/dataset

openmic:/path/to/dataset

(default: None)

Script arguments:

--device str The device to be used eg. cuda:0. (default: cuda)


```
--save-roc          Caculate and save ROC for each instrument (default: False)
--out-dir Path      Sets the output directory (default: None)
--save-metric-hist  Caculate and save histogram for metrics (default: False)
--save-confusion    Caculate and save confusion matrices (default: False)
```

Inference example: script will infer labels and save them to JSON.

```
python3 src/train/run_test.py --dataset-paths data/irmas/test --ckpt models/model.ckpt \
--batch-size 1 --num-workers 1
```

```
100%|██████████| 24/24 [00:05<00:00, 4.04it/s]
Saving json file: /home/██████████/projects/lumen-audio/models/05-06-19-55-50_SlickDelta_efficient_net_v2_s/preds__
_train__irmas_train_val__irmas_test_pred__irmas_test_small.json
```

Figure 1: Console log result

Testing example: script will infer labels and caculate get metrics for the `data/irmas/test` dataset

```
python3 src/train/run_test.py --dataset-paths irmastest:data/irmas/test \
--ckpt models/model.ckpt --batch-size 1 --num-workers 1
```

```
instruments/violin_hamming_distance: 0.1666666865348816
instruments/violin_precision: 0.5555555820465088
instruments/violin_recall: 1.0
precision: 0.18714067339897156
recall: 0.8030303716659546

Saving yaml file: /home/██████████/projects/lumen-audio/models/05-06-19-55-50_SlickDelta_efficient_net_v2_s/metrics_files__train__irmas_train_val__irmas_test_pred__irmas_test_small.yaml
(venv) 15:09 ~/projects/lumen-audio v 1.9 12m10] $
```

Figure 2: Console log result

Testing example with plots: script will infer labels and caculate get metrics for the `data/irmas/test` dataset and generate various plots for better model interpretation (distribution for each metric, confusion matrices for each instrument pair, metrics for each instrument, ROC curve)

```
python3 src/train/run_test.py --dataset-paths irmastest:data/irmas/test \
--ckpt models/model.ckpt --batch-size 1 --num-workers 1 --save-metric-hist \
--save-confusion --save-instrument-metrics --save-roc
```

```
instruments/violin_precision: 0.5555555820465088
instruments/violin_recall: 1.0
precision: 0.18714067339897156
recall: 0.8030303716659546

Saving yaml file: /home/██████████/projects/lumen-audio/models/05-06-19-55-50_SlickDelta_efficient_net_v2_s/metrics_files__train__irmas_train_val__irmas_test_pred__irmas_test_small.yaml
Saving to: /home/██████████/projects/lumen-audio/models/05-06-19-55-50_SlickDelta_efficient_net_v2_s/metrics_hist__train__irmas_train_val__irmas_test_pred__irmas_test_small.png
Saving metric histograms to: /home/██████████/projects/lumen-audio/models/05-06-19-55-50_SlickDelta_efficient_net_v2_s/metric_hist
Saving confusion matrices to: /home/██████████/projects/lumen-audio/models/05-06-19-55-50_SlickDelta_efficient_net_v2_s/conf_matrix
```

Figure 3: Console log result

7 Training

The list of possible train arguments is quite extensive which is why multiple training examples will be shown instead of describing every argument. To get a good feel of what's possible with `src/train/train.py` we suggest you check out the `ConfigDefault` class in `src/inference/run_test.py` file or run `python3 src/train/train.py -h` (you can ignore the `pl.Trainer` argument group).

You can stop the training anytime with `Ctrl + C`. Pressing it once will gracefully shutdown the training (and perform the testing phase). Pressing it twice shows more aggression, which will stop the training immediately.

Relevant enum values

```
AudioTransforms: AST, MEL_SPECTROGRAM, MULTI_SPECTROGRAM, WAV2VEC, MFCC, WAVELET
                  MELCHROWAV
MetricMode: MIN, MAX
ModelInputDataType: WAVEFORM, IMAGE
OptimizeMetric: VAL_HAMMING, VAL_F1
SupportedAugmentations: CONCAT_N_SAMPLES, SUM_N_SAMPLES, BACKGROUND_NOISE
                       TIME_STRETCH, TIME_SHIFT, PITCH, COLOR_NOISE, TIMEINV
                       TIME_MASK, NORM_AFTER_TIME_AUGS, FREQ_MASK, RANDOM_ERASE,
                       RANDOM_PIXELS, SEVEN_BAND_EQ, CLIPPING, SUM_TWO_SAMPLES
SupportedDatasetDirType: irmastrain, irmatest, csv, inference, openmic
SupportedHeads: DEEP_HEAD, ATTENTION_HEAD
SupportedLossFunctions: CROSS_ENTROPY, CROSS_ENTROPY_POS_WEIGHT, FOCAL_LOSS,
                       FOCAL_LOSS_POS_WEIGHT, INSTRUMENT_FAMILY_LOSS
SupportedModels: AST, EFFICIENT_NET_V2_S, EFFICIENT_NET_V2_M, EFFICIENT_NET_V2_L,
                 RESNEXT50_32X4D, RESNEXT101_32X8D, RESNEXT101_64X4D, WAV2VEC,
                 WAV2VEC_CNN, CONVNEXT_TINY, CONVNEXT_SMALL, CONVNEXT_LARGE,
                 CONVNEXT_BASE, MOBILENET_V3_LARGE, MOBNET
SupportedOptimizer: ADAM, ADAMW
SupportedScheduler: ONECYCLE, PLATEAU, AUTO_LR, COSINEANNEALING
```

Recommended models are convolutional neural networks (`EFFICIENT_NET_V2_S` , `CONVNEXT_LARGE` , `MOBILENET_V3_LARGE` ...) with `MEL_SPECTROGRAM` audio transform.

Minimal training example. Trains the whole model:

```
python3 src/train/train.py --train-paths irmastrain:data/irmas/train \
--val-paths irmatest:data/irmas/test --model CONVNEXT_TINY \
--audio-transform MEL_SPECTROGRAM --nofinetune-head
```

Quick training with reduced dataset and batch size:

```
python3 src/train/train.py --train-paths irmastrain:data/irmas/train \
--val-paths irmatest:data/irmas/test --model CONVNEXT_TINY \
--audio-transform MEL_SPECTROGRAM --nofinetune-head --quick
```

Quick training (reduces dataset size and limits batch size to 2):

```
python3 src/train/train.py --train-paths irmastrain:data/irmas/train \
--val-paths irmatest:data/irmas/test --model CONVNEXT_TINY \
--audio-transform MEL_SPECTROGRAM --nofinetune-head
```

Use less resources by setting `--batch-size 1 --num-workers 1`:

```
python3 src/train/train.py --train-paths irmastrain:data/irmas/train \
--val-paths irmatest:data/irmas/test --model CONVNEXT_TINY \
--audio-transform MEL_SPECTROGRAM --nofinetune-head --batch-size 1 --num-workers 1
```

Train only the head of the model:

```
python3 src/train/train.py --train-paths irmastrain:data/irmas/train \
--val-paths irmatest:data/irmas/test --model CONVNEXT_TINY --audio-transform MEL_SPECTROGRAM \
```

```
--head-after classifier --backbone-after classifier
```

Train the head for 5 epochs and then unfreeze the backbone from `layer3.2` onwards:

```
python3 src/train/train.py --train-paths irmastrain:data/irmas/train \
--val-paths irmatest:data/irmas/test --model CONVNEXT_TINY \
--audio-transform MEL_SPECTROGRAM --head-after classifier --finetune_head_epochs 5 \
--backbone-after layer3.2
```

Same as previous command with explicit learning rates. Learning rate changes through time in the following order (-lr-warmup -> -lr -> -lr-onecycle-max -> -lr -> 0):

```
python3 src/train/train.py --model CONVNEXT_TINY --audio-transform MEL_SPECTROGRAM \
--head-after classifier --finetune_head_epochs 5 \
--backbone-after layer3.2 --lr-warmup 5e-5 --lr 1e-5 --lr-onecycle-max 1e-4
```

Add augmentations to training:

```
# you can add --aug-kwarg which overrides augmentation kwargs
# (e.g. --aug-kwarg stretch_factors=0.9,1.1 freq_mask_param=50 sum_n_samples=3)
python3 src/train/train.py --train-paths irmastrain:data/irmas/train \
--val-paths irmatest:data/irmas/test --model CONVNEXT_TINY \
--audio-transform MEL_SPECTROGRAM --augmentations TIME_STRETCH TIME_SHIFT PITCH FREQ_MASK
```

Override early stopping metric:

```
python3 src/train/train.py --train-paths irmastrain:data/irmas/train \
--val-paths irmatest:data/irmas/test --model CONVNEXT_TINY \
--audio-transform MEL_SPECTROGRAM --metric VAL_F1 --metric-mode MAX
```

To list all arguments please run `python3 src/train/train.py -h`.

7.1 Training logs

During the training, the script will create the directory inside `models/05-08-15-11-31_experiment_name` with the following structure:

```
├─ checkpoints
│   └─ my_model.ckpt # model checkpoint
├─ config.yaml      # configuration
├─ events.out.tf     events.1683558698.43a642fdf66e.1976756.0
├─ hparams.yaml     # model hparams (pytorch lightning)
├─ log.txt          # full log file
└─ metrics.csv      # metrics in .csv format
```

8 Tensorboard logs

Tensorboard logging is enabled by default. To see training and validation logs, run the command bellow. Logs should be available in a browser at `http://localhost:6006/`. For more options, check `tensorboard -h`.

```
tensorboard --port 6006 --logdir models/
```

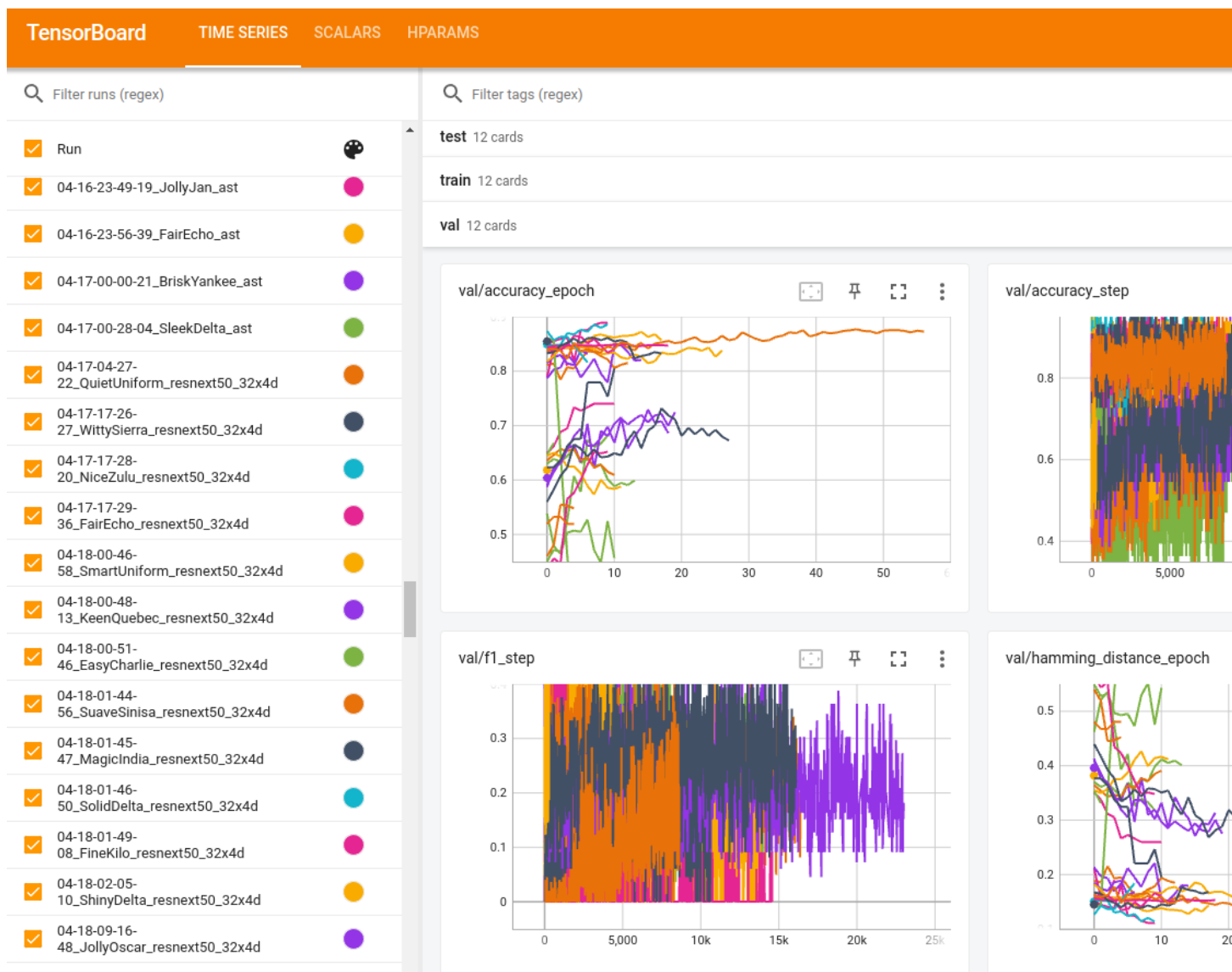


Figure 4: Tensorboard logs

API Reference

Audio prediction API

API Version: 0.1.0

API used for testing and predicting audio data on trained models.

Quick instructions:

1. Make sure you have a .ckpt model inside of the --model-dir directory.
2. Click on GET /models
3. Send a request to get all available models.
4. Copy the model path you want to use for prediction.
5. Click on POST /model/predict-files
6. Paste the model path into the model_ckpt_path field.
7. Choose the audio files you want to predict.
8. Predict labels for the audio files.

INDEX

1. MODEL INFERENCE	3
1.1 POST /model/test-directory-stream	3
1.2 POST /model/test-directory	5
1.3 POST /model/predict-directory-stream	7
1.4 POST /model/predict-directory	8
1.5 POST /model/predict-files	8
2. RESOURCES	10
2.1 GET /models	10
2.2 GET /dataset-types	10

API

1. MODEL INFERENCE

Endpoints used for predicting audio data on trained models. Curl example for multiple audios:

```
curl -X 'POST' \http://localhost:8090/model/predict-files?model_ckpt_path=models%2Fmy_model.ckpt' \-H 'accept: application/json' \-H 'Content-Type: multipart/form-data' \-F 'audio_files=@track_23.wav;type=audio/wav'
```

1.1 POST /model/test-directory-stream

Test Directory Stream

Streams classes of audios from a directory and returns metrics for multiclass classification. Streaming is useful if your directory is large and you don't want to wait for the whole directory to be processed. If you want to process the whole directory at once, use the endpoint test-directory

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
model_ckpt_path	path	

REQUEST BODY - application/json

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
accuracy*	number	
hamming_distance*	number	
f1*	number	
precision*	number	
recall*	number	
instruments/cello_accuracy*	number	
instruments/clarinet_accuracy*	number	
instruments/flute_accuracy*	number	
instruments/acoustic_guitar_accuracy*	number	
instruments/electric_guitar_accuracy*	number	
instruments/organ_accuracy*	number	
instruments/piano_accuracy*	number	
instruments/saxophone_accuracy*	number	
instruments/trumpet_accuracy*	number	
instruments/violin_accuracy*	number	

NAME	TYPE	DESCRIPTION
instruments/human_voice_accuracy*	number	
instruments/cello_hamming_distance*	number	
instruments/clarinet_hamming_distance*	number	
instruments/flute_hamming_distance*	number	
instruments/acoustic_guitar_hamming_distance*	number	
instruments/electric_guitar_hamming_distance*	number	
instruments/organ_hamming_distance*	number	
instruments/piano_hamming_distance*	number	
instruments/saxophone_hamming_distance*	number	
instruments/trumpet_hamming_distance*	number	
instruments/violin_hamming_distance*	number	
instruments/human_voice_hamming_distance*	number	
instruments/cello_f1*	number	
instruments/clarinet_f1*	number	
instruments/flute_f1*	number	
instruments/acoustic_guitar_f1*	number	
instruments/electric_guitar_f1*	number	
instruments/organ_f1*	number	
instruments/piano_f1*	number	
instruments/saxophone_f1*	number	
instruments/trumpet_f1*	number	
instruments/violin_f1*	number	
instruments/human_voice_f1*	number	
instruments/cello_precision*	number	
instruments/clarinet_precision*	number	
instruments/flute_precision*	number	
instruments/acoustic_guitar_precision*	number	
instruments/electric_guitar_precision*	number	
instruments/organ_precision*	number	
instruments/piano_precision*	number	
instruments/saxophone_precision*	number	
instruments/trumpet_precision*	number	
instruments/violin_precision*	number	
instruments/human_voice_precision*	number	
instruments/cello_recall*	number	
instruments/clarinet_recall*	number	
instruments/flute_recall*	number	
instruments/acoustic_guitar_recall*	number	
instruments/electric_guitar_recall*	number	
instruments/organ_recall*	number	
instruments/piano_recall*	number	
instruments/saxophone_recall*	number	

NAME	TYPE	DESCRIPTION
instruments/trumpet_recall*	number	
instruments/violin_recall*	number	
instruments/human_voice_recall*	number	
predictions*	object	

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
detail	array	
loc*	array	
ANY:OF	object	
prop0	string	
prop1	integer	
msg*	string	
type*	string	

1.2 POST /model/test-directory

Test Directory

Infers classes of audios from a directory and returns metrics for multiclass classification.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
model_ckpt_path	path	

REQUEST BODY - application/json

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
accuracy*	number	
hamming_distance*	number	
f1*	number	
precision*	number	
recall*	number	
instruments/cello_accuracy*	number	
instruments/clarinet_accuracy*	number	
instruments/flute_accuracy*	number	

NAME	TYPE	DESCRIPTION
instruments/acoustic_guitar_accuracy*	number	
instruments/electric_guitar_accuracy*	number	
instruments/organ_accuracy*	number	
instruments/piano_accuracy*	number	
instruments/saxophone_accuracy*	number	
instruments/trumpet_accuracy*	number	
instruments/violin_accuracy*	number	
instruments/human_voice_accuracy*	number	
instruments/cello_hamming_distance*	number	
instruments/clarinet_hamming_distance*	number	
instruments/flute_hamming_distance*	number	
instruments/acoustic_guitar_hamming_distance*	number	
instruments/electric_guitar_hamming_distance*	number	
instruments/organ_hamming_distance*	number	
instruments/piano_hamming_distance*	number	
instruments/saxophone_hamming_distance*	number	
instruments/trumpet_hamming_distance*	number	
instruments/violin_hamming_distance*	number	
instruments/human_voice_hamming_distance*	number	
instruments/cello_f1*	number	
instruments/clarinet_f1*	number	
instruments/flute_f1*	number	
instruments/acoustic_guitar_f1*	number	
instruments/electric_guitar_f1*	number	
instruments/organ_f1*	number	
instruments/piano_f1*	number	
instruments/saxophone_f1*	number	
instruments/trumpet_f1*	number	
instruments/violin_f1*	number	
instruments/human_voice_f1*	number	
instruments/cello_precision*	number	
instruments/clarinet_precision*	number	
instruments/flute_precision*	number	
instruments/acoustic_guitar_precision*	number	
instruments/electric_guitar_precision*	number	
instruments/organ_precision*	number	
instruments/piano_precision*	number	
instruments/saxophone_precision*	number	
instruments/trumpet_precision*	number	
instruments/violin_precision*	number	
instruments/human_voice_precision*	number	
instruments/cello_recall*	number	

NAME	TYPE	DESCRIPTION
instruments/clarinet_recall*	number	
instruments/flute_recall*	number	
instruments/acoustic_guitar_recall*	number	
instruments/electric_guitar_recall*	number	
instruments/organ_recall*	number	
instruments/piano_recall*	number	
instruments/saxophone_recall*	number	
instruments/trumpet_recall*	number	
instruments/violin_recall*	number	
instruments/human_voice_recall*	number	
predictions*	object	

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
detail	array	
loc*	array	
ANY:OF	object	
prop0	string	
prop1	integer	
msg*	string	
type*	string	

1.3 POST /model/predict-directory-stream

Predict Directory Stream

Streams classes of audios from a directory. Streaming is useful if your directory is large and you don't want to wait for the whole directory to be processed. If you want to process the whole directory at once, use the endpoint predict-directory. **You don't have to specify the dataset structure type is always inference**

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
model_ckpt_path	path	

REQUEST BODY - application/json

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json		
NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
detail	array	
loc*	array	
ANY:OF	object	
prop0	string	
prop1	integer	
msg*	string	
type*	string	

1.4 POST /model/predict-directory

Predict Directory

Infers classes of audios from a directory.You don't have to specify the dataset structure type is always inference

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
model_ckpt_path	path	

REQUEST BODY - application/json

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json		
NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
detail	array	
loc*	array	
ANY:OF	object	
prop0	string	
prop1	integer	
msg*	string	
type*	string	

1.5 POST /model/predict-files

Predict Files

Infers classes of audios from a list of files.You don't have to specify the dataset structure type is always inference

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
model_ckpt_path	path	

FORM DATA PARAMETERS

NAME	TYPE	DESCRIPTION
audio_files	array of string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
cel*	number	
cla*	number	
flu*	number	
gac*	number	
gel*	number	
org*	number	
pia*	number	
sax*	number	
tru*	number	
vio*	number	
voi*	number	

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
detail	array	
loc*	array	
ANY:OF	object	
prop0	string	
prop1	integer	
msg*	string	
type*	string	

2. RESOURCES

Models and datasets available for inference.

2.1 GET /models

Get Models

Returns paths to all available models on the server. Use these model paths for all POST request predictions. Models are fetched from the directory --model-dir. Only models with the extension *.ckpt are fetched.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

2.2 GET /dataset-types

Get Supported Datasets

Returns dataset types, each dataset type has a strategy for loading labels from a given directory. If you don't have labels or don't care about them, you can use the empty string "" which only loads the audio. If you want to load labels use the following dataset types: ['irmastrain', 'irmastest', 'csv', 'inference', 'openmic']

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json
