# Detekce zvukových událostí

Dokumentace implementace metod a práce s datasetem DESED

Jiří Šeps

Dokumentace pro předmět PDO

16. dubna 2025

## Obsah

# 1 Úvod

Detekce zvukových událostí (Sound Event Detection, SED) je klíčovou disciplínou ve zpracování akustických dat. Cílem této dokumentace je poskytnout přehledný a praktický návod k implementaci, spuštění a vyhodnocení dvou metod pro detekci zvukových událostí, a to v kontextu soutěže **DCASE 2022 Task 4**.

První metodou je výchozí baseline řešení poskytnuté organizátory soutěže DCASE 2022, které slouží jako referenční bod pro další experimenty. Druhá metoda, označovaná jako **FDY-CRNN** (někdy také **FDY-SED**), představuje pokročilé řešení, které bylo navrženo jako účastnický příspěvek do této soutěže a vykazuje výrazně lepší výkonnost.

Obě metody jsou testovány na různých konfiguracích datové sady **DESED**, která slouží jako standardní benchmark pro tento typ úloh. Vyhodnocení probíhá pomocí standardních metrik používaných v rámci DCASE výzev, jako je například *PSDS* (Polyphonic Sound Detection Score).

Tento dokument je strukturován jako návod typu *walkthrough* a má sloužit jako praktický průvodce procesem nastavení prostředí, spuštění metod, trénování modelů a interpretace dosažených výsledků.

# 2 Implementace metod

Tato část popisuje základní kroky pro přípravu pracovního prostředí, klonování repozitářů a konfiguraci potřebnou pro úspěšné spuštění obou metod.

## 2.1 Vývojové prostředí – Visual Studio Code

V rámci tohoto návodu budeme využívat vývojové prostředí **Visual Studio Code (VSC)**, které doporučujeme pro jeho jednoduché ovládání a podporu Python projektů. Lze jej stáhnout z oficiálních stránek: `https://code.visualstudio.com/`

Po otevření projektu ve VSC nainstalujte rozšíření `Python`.

## 2.2 Klonování repozitářů

Obě metody jsou veřejně dostupné na GitHubu. Nejprve si je naklonujeme pomocí příkazového řádku, který lze otevřít pomocí klávesové zkratky **Ctrl** + ˜, nebo navigace do **Terminal** v horní sekci a **New Terminal**

Listing 1: Klonování repozitářů

```
git clone https://github.com/DCASE-REPO/DESED_task.git
cd DESED_task/recipes/dcase2022_task4_baseline

# v jine slozce
git clone https://github.com/frednam93/FDY-SED.git
```

## 2.3 Instalace Pythonu a PyTorch Lightning

Doporučená verze Pythonu je **3.9**. Lze ji stáhnout z: `https://www.python.org/downloads/release/python-390/`

Pro správné fungování obou metod je nutné mít nainstalovaný PyTorch Lightning:

Listing 2: Instalace PyTorch Lightning

```
pip install lightning
```

### 2.4 Vytvoření prostředí přes Conda

Obě metody využívají předpřipravený skript `conda_create_environment.sh`, který obsahuje seznam všech požadovaných knihoven.

Spusťte následující příkaz v příslušném adresáři:

Listing 3: Vytvoření Conda prostředí

```
bash conda_create_environment.sh
```

### 2.5 Nároky na disk

Instalace knihoven, modelů a datasetu může dohromady zabrat přibližně:

- $\sim$ 4–6 GB pro knihovny a prostředí (v závislosti na systému)
- $\sim$ 2–3 GB pro checkpointy modelů a logy

Doporučené volné místo na disku: **minimálně 10 GB**.

### 2.6 Úpravy nekompatibilit – `append` vs `concat`

V novějších verzích knihovny `pandas` je metoda `.append()` označena jako *deprecated*. Pokud při spuštění narazíte na chybu, doporučuje se nahradit následujícím způsobem:

Listing 4: Náhrada deprecated metody

```
# Puvodni kod
df = df.append(new_row, ignore_index=True)

# Novy zapis
df = pd.concat([df, pd.DataFrame([new_row])], ignore_index=True)
```

### 2.7 Úprava: odstranění podpory `unlabeled` datasetu

Obě metody obsahují kód, který předpokládá využití tzv. **unlabeled datasetu**, tedy neanotovaných reálných nahrávek. Tento dataset však není veřejně dostupný a v rámci tohoto návodu jej nebudeme používat. Z tohoto důvodu je nutné ručně upravit kód a zakomentovat příslušné části.

#### DCASE baseline metoda

Otevřete soubor:

```
recipes/dcase2022_task4_baseline/train_sed.py
```

Vyhledejte a zakomentujte následující blok (kolem řádku s názvem ‘UnlabeledSet‘): Nejlépe tento řádek lze najít pomocí klávesové zkratky **Ctrl + F** kde následně můžete napsat požadované slovo nebo výraz.

```
# unlabeled_set = UnlabeledSet(
#     config["data"]["unlabeled_folder"],
#     encoder,
#     pad_to=config["data"]["audio_max_len"],
# )
```

Následně v celém souboru zakomentujte nebo odstraňte všechny další výskyty: - proměnné `unlabeled_set`, - konfigurace jako `config["data"]["unlabeled_folder"]`, - případně jakékoli volání s `dataloaders["unlabeled"]` apod.

### FDY-SED metoda

1. Otevřete soubor:

```
datasets/dataset.py
```

Vyhledejte a zakomentujte celou třídu:

```
# class UnlabeledDataset(Dataset):
#     def __init__(...):
#         ...
#     def __getitem__(self, idx):
#         ...
#     def __len__(self):
#         ...
```

2. Poté otevřete soubor:

```
settings.py
```

Odstraňte nebo zakomentujte jakýkoliv výskyt:
- '**UnlabeledDataset(...)**', - '**unlabeled_loader**', - '**unlabeled_dataset**'
Bez těchto úprav by při spuštění mohlo dojít k chybám, protože 'unlabeled_folder' není definován, nebo soubory neexistují. Po jejich zakomentování budou obě metody plně funkční bez semi-supervised části.

## 3 Dataset DESED

Dataset **DESED** (Domestic Environment Sound Event Detection) je standardní benchmarková sada pro úlohy detekce zvukových událostí. Obsahuje různé typy nahrávek s anotacemi a je součástí výzev DCASE (Detection and Classification of Acoustic Scenes and Events).

### 3.1 Stažení datasetu

Pro stažení části datasetu, označované jako **synthetic**, spusťte skript:

Listing 5: Stažení synthetic části datasetu

```
python generate_dcase_task4_2022.py
```

Tento skript stáhne synteticky generovaná trénovací data – zvukové stopy vytvořené kombinací jednotlivých zvukových událostí z datasetu **AudioSet**.

Pro získání zbylých dat – tedy reálných nahrávek s anotacemi – je potřeba navštívit následující stránku:

- `https://saoyear.github.io/post/downloading-real-waveforms-for-desed/`

Zde si můžete stáhnout:

- **Weak dataset** – reálné nahrávky s *slabými anotacemi* (bez časové lokalizace, pouze informace o přítomnosti tříd).
- **Strong dataset** – reálné nahrávky s *silnými anotacemi* (časové informace začátku a konce událostí).

### 3.2 Rozdělení datasetu a struktura

Dataset je rozdělen do několika částí podle typu anotací a původu nahrávek:

- **Synthetic**:
  - Uměle generovaný dataset vytvořený kombinací událostí z **AudioSet**.
  - Obsahuje silné anotace (časové značky).
  - Používá se hlavně pro počáteční trénování.
- **Weak**:
  - Reálné nahrávky z domácího prostředí.
  - Obsahují pouze slabé anotace (tj. informace o tom, co se v nahrávce nachází, bez časové lokalizace).
- **Strong**:
  - Sada reálných záznamů s plnými časovými anotacemi.
  - Podobný jako *Synthetic*, pouze se jedná o realné nahrávky
- **Unlabelled**:
  - Reálné nahrávky bez anotací, lze použít pro jiné přístupy.
  - V tomto návodu není aktivně využíván.

Typická struktura adresářů po stažení datasetu vypadá následovně:

```
DESED_dataset/
|-- audio/
|   |-- train/
|   |   |-- strong_label_real/
|   |   |-- synthetic21_train/
|   |   |-- weak_16k/
|   |-- validation/
|       |-- synthetic21_validation/
|       |-- validation16_k/
|-- metadata/
    |-- train/
    |-- validation/
```

### 3.3 Požadované místo na disku

Doporučuje se mít vyhrazeno alespoň:

- $\sim$ 10 GB pro synthetic část
- $\sim$ 20 GB pro weak + strong + validation část
- $\sim$ 10 GB rezerva pro logy, modely a další soubory

**Celkem: min. 40 GB volného místa.**

### 3.4 Úprava cest v konfiguraci

Po stažení všech částí datasetu je nutné upravit konfigurační soubor metody, jedná se o soubory config.yaml a default.yaml kde se nastavují cesty k jednotlivým datovým složkám, jako například v FDY-SED:

Listing 6: Uprava cest k datasetu FDY-SED

```
dataset:
    weak_folder: "/cesta/k/DESED_dataset/audio/train/weak/"
    weak_tsv: "/cesta/k/DESED_dataset/metadata/weak.tsv"
    strong_folder: "/cesta/k/DESED_dataset/audio/train/strong/"
    strong_tsv: "/cesta/k/DESED_dataset/metadata/strong.tsv"

synth_dataset:
    synth_train_folder: "/cesta/k/DESED_dataset/audio/train/
       synthetic/"
    synth_tsv: "/cesta/k/DESED_dataset/metadata/synthetic.tsv"
```

nebo v DCASE metodě:

Listing 7: Uprava cest k datasetu DCASE

```
data:
    weak_folder: "/cesta/k/DESED_dataset/audio/train/weak/"
    weak_tsv: "/cesta/k/DESED_dataset/metadata/weak.tsv"
    strong_folder: "/cesta/k/DESED_dataset/audio/train/strong/"
    strong_tsv: "/cesta/k/DESED_dataset/metadata/strong.tsv"
    synth_folder: "/cesta/k/DESED_dataset/audio/train/synthetic/"
    synth_tsv: "/cesta/k/DESED_dataset/metadata/synthetic.tsv"
```

Cesty je třeba přizpůsobit vaší lokální složce, do které jste dataset stáhli.

## 4 Spuštění metod

V této části si ukážeme, jak spustit trénování modelů u obou metod pomocí terminálu ve Visual Studio Code, jaké jsou hardwarové požadavky a jaké existují režimy pro testování a ladění.

### 4.1 Spuštění skriptů

Po otevření složky s danou metodou ve Visual Studio Code spusťte integrovaný terminál (‘Ctrl + ˜‘) a zadejte:

**DCASE baseline metoda**

Listing 8: Spuštění trénování DCASE baseline

```
python train_sed.py
```

nebo

Listing 9: Spuštění trénování DCASE baseline

```
python train_sed.py --strong_real
```

**FDY-SED metoda**

Listing 10: Spuštění trénování FDY-SED

```
python main.py
```

Oba skripty akceptují řadu parametrů pro přizpůsobení běhu modelu, viz níže.

### 4.2 Hardwarové a softwarové požadavky

Pro trénink modelů na datasetu DESED jsou doporučeny následující systémové parametry:

- **GPU:** alespoň 1× NVIDIA GPU (doporučeno 6+ GB VRAM, např. RTX 2060/3060 nebo vyšší)
- **RAM:** 16 GB
- **CPU:** min. 4 jádra
- **Disk:** min. 40 GB volného místa
- **OS:** Linux nebo Windows 10/11
- **Python:** verze 3.8 až 3.10

Trénink na CPU je možný, ale výrazně pomalejší.

### 4.3 Dostupné parametry při spuštění

Při spouštění DCASE metody můžete dále využít tyto parametry

- `-conf_file`: Cesta ke konfiguračnímu souboru (YAML), např. `confs/default.yaml`.
- `-log_dir`: Složka pro ukládání logů, váhových souborů a TensorBoard výstupů. Výchozí: `./exp/2022_baseline`.
- `-strong_real`: Aktivuje použití reálně anotovaných (strong) nahrávek během trénování.
- `-resume_from_checkpoint`: Umožní obnovit trénování z uloženého modelu ve formátu `.ckpt`.

- `-test_from_checkpoint`: Spustí testování na základě uloženého modelu bez dalšího trénování.
- `-eval_from_checkpoint`: Vyhodnotí daný model pomocí validační sady.
- `-gpus`: Specifikace počtu GPU (např. `-gpus 1`) nebo výběr konkrétního zařízení (`"0"` apod.).
- `-fast_dev_run`: Aktivuje ladicí režim (debugging), který spustí pouze několik batchí a epoch pro rychlé otestování správné funkčnosti kódu.

Příklad spuštění s více parametry:

```
python train_sed.py --continue_from_checkpoint --strong_real
```

### 4.4 Testovací a ladicí režim (debugging)

FDY-SED metoda má možnost spouštět model v tzv. **testovacím** nebo **debug** režimu. Tyto režimy se nastavují v konfiguračních souborech YAML:

- `test_only: true` – metoda pouze otestuje model bez trénování
- `debug: true` – aktivuje zrychlené ladění s malým počtem batchí a bez ukládání modelu

Příklad části konfigurace v souboru `conf/train.yaml`:

```
training:
  debug: true
  test_only: false
```

Tyto režimy jsou ideální pro ověření funkčnosti kódu, integrity datasetu nebo pro rychlé spuštění bez dlouhého čekání.

## 5 Trénování

V této části si projdeme, jak probíhá samotné trénování modelů pro detekci zvukových událostí, jak nastavit potřebné parametry, kde hledat výstupy a jak interpretovat dosažené výsledky pomocí běžně používaných metrik.

### 5.1 Nastavení trénování

Parametry trénování jsou uloženy v konfiguračních souborech ve formátu `.yaml` – obvykle např. `conf/train.yaml`, `confs/default.yaml` nebo `config.yaml` v závislosti na metodě.

Příklady parametrů v YAML souboru:

```
training:
  epochs: 100
  batch_size: 32
  learning_rate: 0.001
  test_only: false
  debug: false
```

Doporučuje se začít s menším počtem epoch a aktivovat debug mód (`debug: true`) pro ověření, že vše funguje správně.

### 5.2 Popis jednotlivých parametrů

- `epochs` – kolikrát model projde celý trénovací dataset (více epoch = delší, ale přesnější trénink).
- `batch_size` – počet vzorků zpracovaných najednou (větší hodnoty jsou rychlejší, ale náročnější na paměť).
- `learning_rate` – určuje, jak rychle se model učí. Příliš vysoká může způsobit nestabilitu, příliš nízká zase pomalé učení.
- `debug` – aktivuje „zkušební“ režim s minimálním množstvím dat.
- `test_only` – pokud je nastaveno na `true`, model se pouze otestuje, netrénuje.

### 5.3 Ukládání modelu a kontrola průběhu

Během trénování se průběžně ukládají tzv. **checkpointy**, tedy váhové soubory modelu a výsledky trénování. Tyto soubory najdete ve složce definované v parametru `log_dir`, např.:

```
exp/2022_baseline/version_0/epoch=15-step=1234.ckpt
```

### 5.4 Vyhodnocení výsledků – metriky

Po tréninku se model vyhodnocuje pomocí několika standardních metrik, které se běžně používají v DCASE výzvách. Níže je jejich stručné vysvětlení, zaměřené na praktické pochopení:

- **F1 skóre (collar-based)** – klasická přesnost modelu kombinující přesnost (precision) a úplnost (recall), kde se toleruje určitá odchylka v čase (tzv. „collar“).
- **F1 skóre (event-based)** – hodnotí přesnost detekce jednotlivých událostí v čase. Přesnější, ale přísnější metrika než collar-based.
- **PSDS (Polyphonic Sound Detection Score)** – hlavní metrika soutěže DCASE. Zohledňuje celkovou kvalitu detekce napříč různými prahy (srovnatelné s „plošným skóre“ jako plocha pod křivkou **(AUC)**). Čím vyšší hodnota (0–1), tím lepší výkon.
- **Error Rate (ER)** – jednoduchá míra chybovosti, ukazuje kolik událostí model rozpoznal špatně. Nižší hodnota je lepší.

Výstupy metrik se vypisují buď v konzoli, nebo jsou uloženy v `.csv` nebo `.txt` souborech v log složce. Pro účely srovnání a ladění doporučujeme zaznamenávat:

- nejlepší dosažené **PSDS**
- **F1 score (event-based)**

## 6 Ukázka výsledků

V této části uvádíme praktické ukázky výstupů z obou metod. Obrázky zobrazují například průběh trénování, predikce detekovaných událostí nebo vizualizaci výstupu PSDS skóre.

```
date & time of start is : 2025-03-25 23:08:43
torch version is: 2.5.1+cu121
number of GPUs: 1
device: cuda:0
Total Trainable Params: 11.061 M
   training starts!
[Epc 1] tt: 0.874, cl_st: 0.576, cl_wk: 0.596, cn_st: 0.003, cn_wk: 0.000, st_vl: 0.392, t_vl: 0.198, t: 170s
[Epc 2] tt: 0.542, cl_st: 0.328, cl_wk: 0.426, cn_st: 0.033, cn_wk: 0.000, st_vl: 0.000, t_vl: 0.385, t: 169s
[Epc 3] tt: 0.433, cl_st: 0.238, cl_wk: 0.385, cn_st: 0.039, cn_wk: 0.000, st_vl: 0.000, t_vl: 0.286, t: 169s
[Epc 4] tt: 0.422, cl_st: 0.231, cl_wk: 0.380, cn_st: 0.022, cn_wk: 0.000, st_vl: 0.002, t_vl: 0.303, t: 168s
[Epc 5] tt: 0.415, cl_st: 0.227, cl_wk: 0.375, cn_st: 0.014, cn_wk: 0.000, st_vl: 0.086, t_vl: 0.256, t: 169s
[Epc 6] tt: 0.407, cl_st: 0.222, cl_wk: 0.369, cn_st: 0.010, cn_wk: 0.000, st_vl: 0.081, t_vl: 0.205, t: 167s
[Epc 7] tt: 0.407, cl_st: 0.223, cl_wk: 0.366, cn_st: 0.008, cn_wk: 0.000, st_vl: 0.049, t_vl: 0.166, t: 167s
[Epc 8] tt: 0.396, cl_st: 0.215, cl_wk: 0.361, cn_st: 0.008, cn_wk: 0.000, st_vl: 0.048, t_vl: 0.185, t: 166s
[Epc 9] tt: 0.390, cl_st: 0.213, cl_wk: 0.352, cn_st: 0.008, cn_wk: 0.000, st_vl: 0.380, t_vl: 0.124, t: 167s
[Epc 10] tt: 0.372, cl_st: 0.202, cl_wk: 0.336, cn_st: 0.010, cn_wk: 0.000, st_vl: 0.247, t_vl: 0.161, t: 166s
[Epc 11] tt: 0.361, cl_st: 0.197, cl_wk: 0.323, cn_st: 0.012, cn_wk: 0.000, st_vl: 0.333, t_vl: 0.086, t: 167s
[Epc 12] tt: 0.358, cl_st: 0.200, cl_wk: 0.311, cn_st: 0.013, cn_wk: 0.000, st_vl: 0.315, t_vl: 0.145, t: 166s
[Epc 13] tt: 0.353, cl_st: 0.198, cl_wk: 0.304, cn_st: 0.013, cn_wk: 0.000, st_vl: 0.387, t_vl: 0.169, t: 166s
[Epc 14] tt: 0.345, cl_st: 0.194, cl_wk: 0.295, cn_st: 0.013, cn_wk: 0.000, st_vl: 0.394, t_vl: 0.193, t: 167s
[Epc 15] tt: 0.332, cl_st: 0.186, cl_wk: 0.284, cn_st: 0.013, cn_wk: 0.000, st_vl: 0.455, t_vl: 0.156, t: 167s
[Epc 16] tt: 0.333, cl_st: 0.187, cl_wk: 0.281, cn_st: 0.013, cn_wk: 0.000, st_vl: 0.486, t_vl: 0.188, t: 166s
[Epc 17] tt: 0.328, cl_st: 0.186, cl_wk: 0.272, cn_st: 0.012, cn_wk: 0.000, st_vl: 0.597, t_vl: 0.223, t: 166s
[Epc 18] tt: 0.322, cl_st: 0.182, cl_wk: 0.269, cn_st: 0.012, cn_wk: 0.000, st_vl: 0.673, t_vl: 0.235, t: 166s
[Epc 19] tt: 0.316, cl_st: 0.180, cl_wk: 0.260, cn_st: 0.011, cn_wk: 0.000, st_vl: 0.589, t_vl: 0.263, t: 167s
[Epc 20] tt: 0.309, cl_st: 0.179, cl_wk: 0.246, cn_st: 0.011, cn_wk: 0.000, st_vl: 0.766, t_vl: 0.292, t: 166s
[Epc 21] tt: 0.299, cl_st: 0.173, cl_wk: 0.237, cn_st: 0.010, cn_wk: 0.000, st_vl: 0.697, t_vl: 0.364, t: 166s
[Epc 22] tt: 0.305, cl_st: 0.175, cl_wk: 0.245, cn_st: 0.009, cn_wk: 0.000, st_vl: 0.736, t_vl: 0.398, t: 166s
[Epc 23] tt: 0.300, cl_st: 0.172, cl_wk: 0.240, cn_st: 0.009, cn_wk: 0.000, st_vl: 0.753, t_vl: 0.472, t: 166s
[Epc 24] tt: 0.285, cl_st: 0.167, cl_wk: 0.219, cn_st: 0.009, cn_wk: 0.000, st_vl: 0.781, t_vl: 0.485, t: 166s
[Epc 25] tt: 0.292, cl_st: 0.170, cl_wk: 0.226, cn_st: 0.008, cn_wk: 0.000, st_vl: 0.814, t_vl: 0.561, t: 166s
[Epc 26] tt: 0.290, cl_st: 0.170, cl_wk: 0.219, cn_st: 0.008, cn_wk: 0.000, st_vl: 0.852, t_vl: 0.559, t: 166s
[Epc 27] tt: 0.281, cl_st: 0.166, cl_wk: 0.211, cn_st: 0.007, cn_wk: 0.000, st_vl: 0.947, t_vl: 0.728, t: 166s
[Epc 28] tt: 0.291, cl_st: 0.173, cl_wk: 0.213, cn_st: 0.008, cn_wk: 0.000, st_vl: 0.906, t_vl: 0.821, t: 166s
[Epc 29] tt: 0.282, cl_st: 0.164, cl_wk: 0.215, cn_st: 0.007, cn_wk: 0.000, st_vl: 0.925, t_vl: 0.872, t: 166s
[Epc 30] tt: 0.290, cl_st: 0.171, cl_wk: 0.214, cn_st: 0.007, cn_wk: 0.000, st_vl: 0.913, t_vl: 0.857, t: 166s
```

Obrázek 1: Průběh statistik po jednotlivých epochách v FDY-SED.

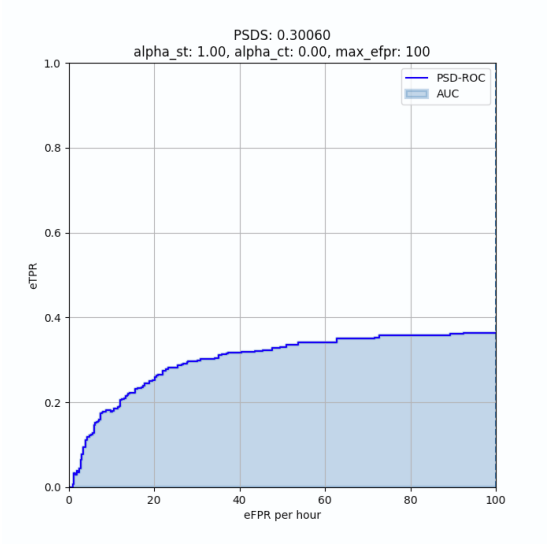
```
Event based metrics (onset-offset)
========================================
  Evaluated length                : 10459.12 sec
  Evaluated files                 : 1168
  Evaluate onset                  : True
  Evaluate offset                 : True
  T collar                        : 200.00 ms
  Offset (length)                 : 20.00 %

  Overall metrics (micro-average)
  ========================================
  F-measure
    F-measure (F1)                : 41.52 %
    Precision                     : 54.04 %
    Recall                        : 33.71 %
  Error rate
    Error rate (ER)               : 0.93
    Substitution rate             : 0.02
    Deletion rate                 : 0.65
    Insertion rate                : 0.27

  Class-wise average metrics (macro-average)
  ========================================
  F-measure
    F-measure (F1)                : 36.07 %
    Precision                     : 47.97 %
    Recall                        : 33.16 %
  Error rate
    Error rate (ER)               : 1.04
    Deletion rate                 : 0.67
    Insertion rate                : 0.37


  Class-wise metrics
  ========================================
    Event label  | Nref  | Nsys  | F      | Pre    | Rec    | ER     | Del    | Ins    |
    -------------|-------|-------|--------|--------|--------|--------|--------|--------|
    Cat          | 341   | 178   | 33.9%  | 49.4%  | 25.8%  | 1.01   | 0.74   | 0.26   |
    Vacuum_cle.. | 92    | 51    | 40.6%  | 56.9%  | 31.5%  | 0.92   | 0.68   | 0.24   |
    Speech       | 1752  | 1543  | 54.4%  | 58.1%  | 51.1%  | 0.86   | 0.49   | 0.37   |
    Dog          | 570   | 53    | 6.4%   | 37.7%  | 3.5%   | 1.02   | 0.96   | 0.06   |
    Electric_s.. | 65    | 67    | 54.5%  | 53.7%  | 55.4%  | 0.92   | 0.45   | 0.48   |
    Blender      | 94    | 95    | 37.0%  | 36.8%  | 37.2%  | 1.27   | 0.63   | 0.64   |
    Running_wa.. | 237   | 184   | 34.2%  | 39.1%  | 30.4%  | 1.17   | 0.70   | 0.47   |
    Alarm_bell.. | 420   | 293   | 50.5%  | 61.4%  | 42.9%  | 0.84   | 0.57   | 0.27   |
    Dishes       | 559   | 42    | 7.0%   | 50.0%  | 3.8%   | 1.00   | 0.96   | 0.04   |
    Frying       | 94    | 129   | 42.2%  | 36.4%  | 50.0%  | 1.37   | 0.50   | 0.87   |
```

(a) Příklad výstupu event_f1.txt v DCASE metodě .

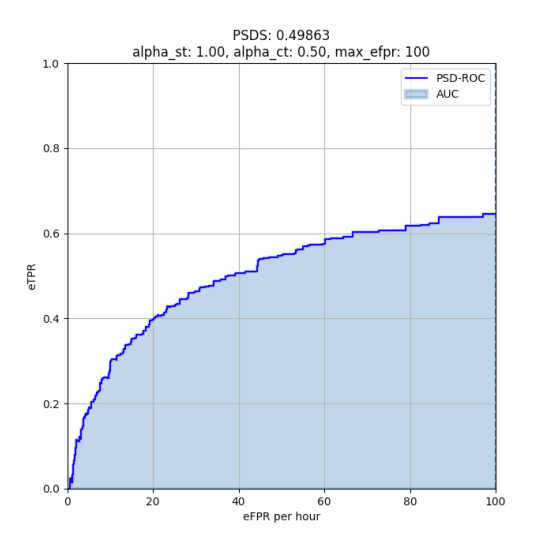
```
Event based metrics (onset-offset)
========================================
  Evaluated length                : 10459.12 sec
  Evaluated files                 : 1168
  Evaluate onset                  : True
  Evaluate offset                 : True
  T collar                        : 200.00 ms
  Offset (length)                 : 20.00 %

  Overall metrics (micro-average)
  ========================================
  F-measure
    F-measure (F1)                : 47.70 %
    Precision                     : 53.16 %
    Recall                        : 43.25 %
  Error rate
    Error rate (ER)               : 0.93
    Substitution rate             : 0.02
    Deletion rate                 : 0.55
    Insertion rate                : 0.36

  Class-wise average metrics (macro-average)
  ========================================
  F-measure
    F-measure (F1)                : 47.60 %
    Precision                     : 50.24 %
    Recall                        : 46.23 %
  Error rate
    Error rate (ER)               : 1.01
    Deletion rate                 : 0.54
    Insertion rate                : 0.47


  Class-wise metrics
  ========================================
    Event label  | Nref  | Nsys  | F      | Pre    | Rec    | ER     | Del    | Ins    |
    -------------|-------|-------|--------|--------|--------|--------|--------|--------|
    Vacuum_cle.. | 92    | 80    | 65.1%  | 70.0%  | 60.9%  | 0.65   | 0.39   | 0.26   |
    Cat          | 341   | 333   | 47.5%  | 48.0%  | 46.9%  | 1.04   | 0.53   | 0.51   |
    Dog          | 570   | 479   | 28.2%  | 30.9%  | 26.0%  | 1.32   | 0.74   | 0.58   |
    Alarm_bell.. | 420   | 323   | 47.6%  | 54.8%  | 42.1%  | 0.93   | 0.58   | 0.35   |
    Speech       | 1752  | 1360  | 57.8%  | 66.2%  | 51.4%  | 0.75   | 0.49   | 0.26   |
    Frying       | 94    | 120   | 46.7%  | 41.7%  | 53.2%  | 1.21   | 0.47   | 0.74   |
    Blender      | 94    | 126   | 50.0%  | 43.7%  | 58.5%  | 1.17   | 0.41   | 0.76   |
    Electric_s.. | 65    | 67    | 57.6%  | 56.7%  | 58.5%  | 0.86   | 0.42   | 0.45   |
    Running_wa.. | 237   | 183   | 41.9%  | 48.1%  | 37.1%  | 1.03   | 0.63   | 0.40   |
    Dishes       | 559   | 366   | 33.5%  | 42.3%  | 27.7%  | 1.10   | 0.72   | 0.38   |
```

(b) Příklad výstupu event_f1.txt v FDY-SED metodě.

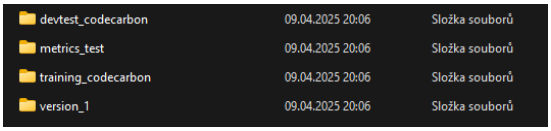
Obrázek 2: Porovnání f1 skóre obou metod.



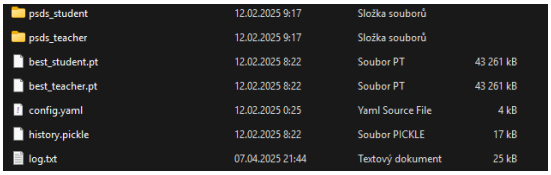

(a) PSDS1




(b) PSDS2




(c) Vzhled složky po natrénovaní DCASE




(d) Vzhled složky po natrénovaní FDY-SED

Obrázek 3: Porovnání PSDS a vzhledu složek po natrénovaní

# 7 Závěr

Tato dokumentace shrnuje proces implementace dvou metod SED nad datasetem DESED. Byly popsány kroky od přípravy prostředí až po analýzu výsledků. Pro detailnější průvodce nebo specifikace těchto metod doporučujeme si přečíst jejich vědecké dokumentace.