# YOHO model for Audio Segmentation and Sound Event Detection

Davide Capone [SM3500601] Enrico Stefanel [SM3500554]
`{davide.capone, enrico.stefanel}@studenti.units.it`

Data Science and Scientific Computing Master's Course
Department of Mathematics and Geosciences
University of Trieste

A.Y. 2023–2024

# Contents

Introduction

YOHO model

Work done and results

Conclusions

# Audio Segmentation and Sound Event Detection

The goal of automatic sound event detection (SED) methods is to recognize what is happening in an audio signal and when it is happening[1]. In practice, the goal is to recognize at what temporal instances different sounds are active within an audio signal. An example of sound event detection is presented below.

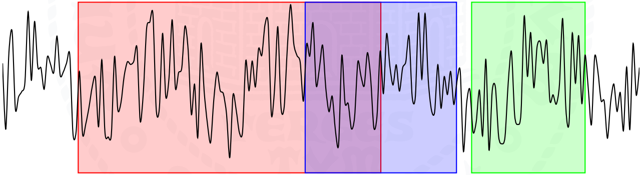


Figure 1: Event Detection in an audio track.

[1] **Mesaros2021SoundED**.

## Datasets

Common datasets for Audio Segmentation and Sound Event Detection problems are:

- **TUT Sound Event Detection**: primarily consists of street recordings with traffic and other activity, with audio examples of 2.56 s and a total size of approximately 1.5 h. It has six unique audio classes – Brakes Squeaking, Car, Children, Large Vehicle, People Speaking, and People Walking;
- **Urban-SED**: purely synthetic dataset, with audio example of 10 s and a total size of about 30 h. It has ten unique audio classes – Air Conditioner, Car Horn, Children Playing, Dog Bark, Drilling, Engine Idling, Gun Shot, Jackhammer, Siren, and Street Music.

The first dataset is too small to train a Neural Network model and requires use of augmentation techniques (we used **SpecAugment**[2]).

[2] **park19e_interspeech**.

## Metrics

A popular toolbox for Sound Event Detection models evaluation is **SED Eval**[3].

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F}_1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where TP, FP and FN are respectively (for each audio segment):

- the ground truth and system output both indicate an event to be active;
- the ground truth indicates an event to be inactive, but the system as active;
- the ground truth indicates an event to be active, but the system as inactive.

[3] **app6060162**.

# YOHO model

Presented in 2021[4]...

[4] **Venkatesh_2022**.

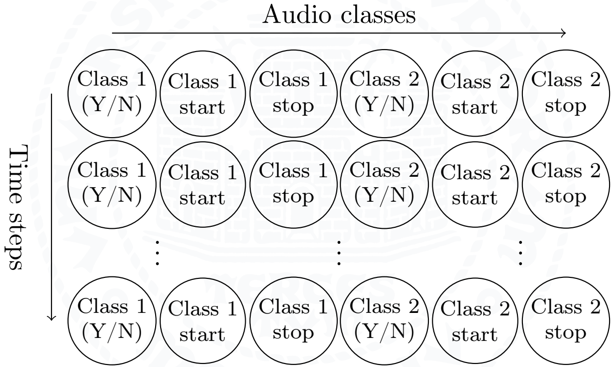
# Input shape

. . .

# Network Architecture

. . .

# Output shape



Audio classes

Time steps

| | | | | | |
|---|---|---|---|---|---|
| Class 1 (Y/N) | Class 1 start | Class 1 stop | Class 2 (Y/N) | Class 2 start | Class 2 stop |
| Class 1 (Y/N) | Class 1 start | Class 1 stop | Class 2 (Y/N) | Class 2 start | Class 2 stop |
| ⋮ | | ⋮ | | ⋮ | |
| Class 1 (Y/N) | Class 1 start | Class 1 stop | Class 2 (Y/N) | Class 2 start | Class 2 stop |

Figure 2: The YOHO output shape.

## Loss Function

$$\mathcal{L}_c(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + \\ (\hat{y}_2 - y_2)^2 + (\hat{y}_3 - y_3)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2, & \text{if } y_1 = 0 \end{cases}$$

where $y$ and $\hat{y}$ are the ground-truth and predictions respectively. $y_1 = 1$ if the acoustic class is present and $y_1 = 0$ if the class is absent. $y_2$ and $y_3$, which are the start and endpoints for each acoustic class are considered only if $y = 1$. In other words, $(\hat{y}_1 - y_1)^2$ corresponds to **the classification loss** and $(\hat{y}_2 - y_2)^2 + (\hat{y}_3 - y_3)^2$ corresponds to **the regression loss**.

# Other Details

. . .

# Implementation challenges

Starting from the original paper, we implemented the system using PyTorch[5], writing the code keeping in mind that it had to be **clear** and permit **reproducible tests**.

```
$ python3 -m yoho.train --help
usage: train.py [-h] [--name NAME] [--epochs EPOCHS] [--batch-size BATCH_SIZE] [--cosine-annealing]
[--autocast] [--spec-augment]

options:
  -h, --help            show this help message and exit
  --name NAME           The name of the model
  --epochs EPOCHS       The number of epochs to train the model
  --batch-size BATCH_SIZE
                        The batch size for training the model
  --cosine-annealing    Use cosine annealing learning rate scheduler
  --autocast            Use autocast to reduce memory usage
  --spec-augment        Augment the training data using SpecAugment
```

Listing 1: Training script parameters

We used ORFEO[6] computational resources for the trainings of the models.

[5] All the code is available at https://github.com/enstit/YOHO24.

[6] https://www.areasciencepark.it/piattaforme-tecnologiche/data-center-orfeo/

# Training results

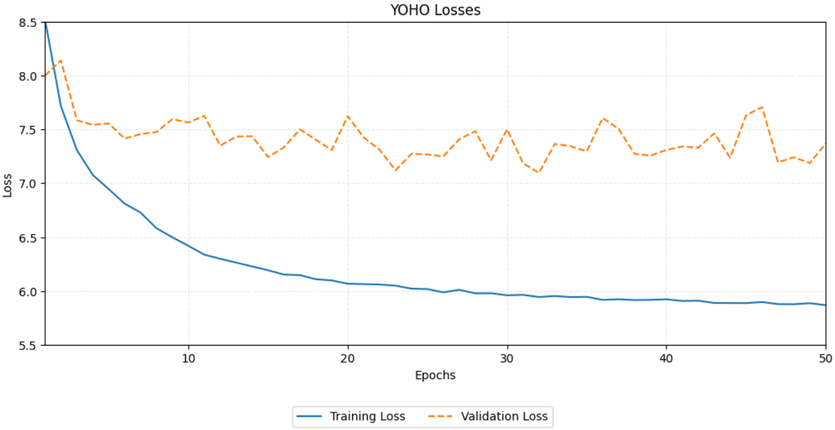


Figure 3: Training and validation loss for YOHO model on UrbanSED dataset.

# Conclusions

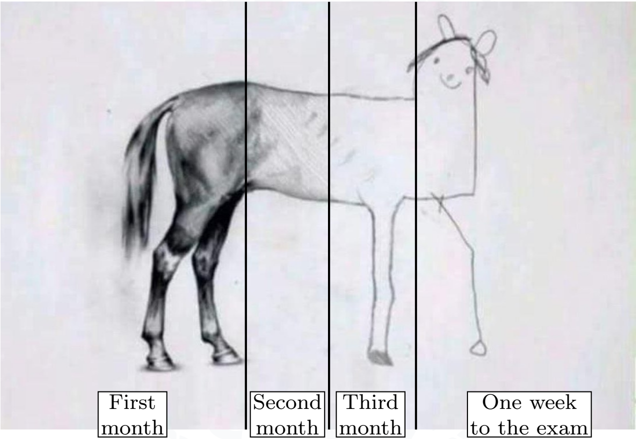


Figure 4: The roadmap of our journey.

# Conclusions

But, after all. . .

*It's all about the journey, not the destination.*

Thank you for your attention.