Speech-YALE
(**Y**ou **A**ren’t **L**istening to **E**verything)

A small FillerWordsRemoval approach

Caruso Gaetano
Politecnico di Torino
s317746@studenti.polito.it

Caruso Antonio
Politecnico di Torino
s313443@studenti.polito.it

Goal and requirements

- Machine Learning model able to detect filler words in an audio file, so that they can be later removed during a post-processing phase
  - They can be proper words or simply sounds:
    - Uh, Um, … or Music, Laughter, ...
- Filler words must be classified and localized
  - Need both class and timestamp predictions
- Model must be able to run on low powered devices
  - Desktops without GPU, laptop, mobile devices

Related works

- Not many works in literature, either too simple or too complex:
  - Classification-only tasks
    - Provide no bounding box information
    - E.g. keyword spotting
  - Transformers-based architectures
    - Very good result quality
    - Computationally heavy, require powerful devices

An intermediate approach

- Speech-YOLO: inspired by popular object detection algorithm YOLO
  - Adapt the algorithm to be applied to audio domain
  - Requires previous features extraction
- Uses a CNN (VGG-19) instead of transformers
  - Lighter on computational resources
- Performs both classification and localization of single words
  - Audio signal previously split in clips
- However, designed for common words, not for fillers

Regular words vs fillers

- Fillers are usually shorter than regular words
  - Difficult to predict bounding box accurately
- Fillers are less structured than words
  - Difficult to detect common features for each class
- Filler word removal is a less common task
  - Finding appropriate dataset is difficult

The Podcast Fillers Dataset

- Specifically designed for filler words detection
- 88.000 annotated clips with length of 1s extracted from full English podcast episodes
- Already split in training, validation and test subsets
- Several event classes (Uh, Um, You know, Like, Words, Repetitions, Laughter, Music, …)
  - Also provides condensed class dictionary (used in this project): Breath, Laughter, Music, Uh, Um, Words
- Labels contain many fields
  - We only used class and timestamps (start and end converted to center and delta for simplicity)

Podcast Fillers limitations

- **Drawback 1**: no negative class training samples
  - i.e. absence of ‘*Nonfiller*’ class
  - Not required in the original project witch led to the creation of this dataset, due to its particular architecture
- **Drawback 2**: all clips contain an event which is always centered on the clip center (event center is always at 0.5s)
  - Bad when performing bounding box regression, because the model can’t generalize to real data, in which the center of the event may be located in an arbitrary position inside the clip
  - Not a problem in the original project, since timestamps are determined by VAD + ASR and only a classifier was trained (i.e. no regressor has to be trained on these data)

Another dataset: LibriSpeech ASR corpus

- Contains recordings from audiobooks
  - Good quality audio
  - Free or almost free of filler words
  - Ideal for building negative class samples
  - Many negative class clips can be extracted

Mixing the datasets

- **Idea**: use positive class clips from Podcast Fillers and negative class clips from LibriSpeech
- Proportion 70% negatives / 30% positives
  - Reflects plausible proportion in real spoken speech

Problems of mixing datasets

- Audio clips from distinct datasets show different features
  - E.g. quality, pronunciation, ambient noise…
- Model may learn to “split” the datasets instead of learning useful features
- Try data augmentation to make data as homogeneous as possible
  - Still, the model does not generalize correctly to new data
- Extracting negative class clips more carefully, e.g. removing silence, does not solve the problem

The single adapted dataset

- **Idea**: extract non-fillers from Podcast Fillers full episodes
- Leverage VAD annotations already available with original dataset to discard silent intervals
- Choose only clips which do not intersect with any filler event
- Re-extract also filler clips from full episodes files by applying random offset in interval [-0.45, 0.45] to remove any constraints on the position of the event center
  - Otherwise, on-the-fly data augmentation needed directly on spectrogram
  - Requires filling empty part of spectrogram with fictitious data
  - Reduction of generalization capabilities
- (about) 50% / 50% proportion of positive and negative classes

Data loading process

- Apply on-the-fly data augmentation on training samples
  - Time stretching, volume scaling, pitch shifting, noise addition
  - Transformations are applied anytime the clip is read from disk, each one with a 50% chance
- Generate dB-Mel-spectrogram
  - Window length = 512 samples, #Mel bins = 128, Hop size = ½ * Window length
  - Values in dB normalized into [0, 1] interval
  - Image resized to 224x224 size
- Load mini-batches with 64 elements each

The model architecture

- Several architecture families considered:
  - ResNet
    - **ResNet-8**
    - **ResNet-18**
    - **ResNet-34**
  - MobileNet
    - ~~MobileNet-v2~~
    - **MobileNet-v3**
  - VGG
    - ~~VGG-19~~
- MobileNet-v2 and VGG-19 revealed, respectively, too slow and too complex to train
- Rationale behind this choice: relatively low complexity, which allows using these models on less powerful machines, too and training simplicity

Classifier and regressor

- All models are equipped with separate classifier and regressor 'heads'
  - Different weights and architecture, i.e. nr. of layers, may be needed, since the two tasks are quite different
  - Regression is usually more complex than classification
- Both implemented with fully connected layers
- Classifier predicts the event class
- Regressor predicts the bounding box coordinates (center and delta)

Loss function

- Different tasks require different loss functions
- **Regression**
  - MSE loss was considered at first for both bounding box coordinates, but we observed that Smooth-L1 loss performed better
    - Probably because the difference between actual and predicted values is very small (both bounding box coordinates are, usually < 1)
  - A third regression ‘coherence’ contribution was also considered, which tries to promote the exact match between the predicted and actual values of the bounding box start and end points (using Smooth-L1, again)
- **Classification**
  - Cross entropy loss
  - Classes have unbalanced nr. of samples, but they’re still equally relevant
    - Weight each class contribution to the loss with the inverse of its occurrences nr., so that even smaller classes gain more importance in the overall loss

The overall loss

- **Question**: how to combine all contributes?
- Classification and regression use different loss types
  - Scale can be considerably different
  - Some of the contributes may be dominant above the others
- Sum all the loss portions, weighting each contribution with a λ coefficient
  - How to choose the value of each λ coefficient?

Choosing λ values

- Start with $\lambda_{center} = 50$ and $\lambda_{delta} = \lambda_{coherence} = 25$
  - Bring classification and regression loss approximately to the same scale
  - Try assigning more importance to the center prediction, since it's the task the model seems to have more trouble with

Dynamic λ weights

- Choosing right values for λ is difficult
- Our initial values are not necessarily correct or the best
- **Idea**: let the model itself learn the best coefficients for each loss contribution
  - Make all λ trainable parameters

Learning rate, optimizer, scheduler

- Optimizer:
  - Adam optimizer
- Learning rate:
  - 10^{-2} → loss diverges
  - 10^{-3} → loss oscillations
  - 10^{-4} → loss converges
- Scheduler:
  - **Idea**: use variable learning rate, making the training process faster
  - OneCycleLR scheduler
  - Start with higher learning rate (10 times the base value) and reduce it gradually to the base value

Strengths of chosen network architectures

- Different *ResNet* versions
  - *ResNet-8*, *ResNet-18*, *ResNet-34*
  - Skip connections reduce vanishing gradient risk
  - Skip connections prevent the model from becoming too complex, reducing the risk of overfitting
- *MobileNet-v3 (large)*
  - Thought for weak devices without GPU
  - Mainly built for mobile devices
- Both families proved to be fast for both training and inference also on low-powered devices

Training process

- Training from scratch was required
  - Pre-trained models are available, but they are trained on images
  - Spectrogram audio features have likely nothing in common with features learned on regular images
  - Fine tuning is thus not a viable option

Evaluation metrics

- Need to evaluate:
  - **Classification**
    - Predicted class
  - **Regression**
    - Bounding box coordinates
  - **Combination of both aspects**
    - Predicted class correctness + bounding box constraints satisfaction

Classification metrics

- Accuracy
  - # of correct class predictions over total predictions
- Precision (per class)
  - # of true positives for a certain class over # of predictions for that class
- Recall (per class)
  - # of true positives for a certain class over # of actual occurrences of that class
- F1-Score (per class)
  - Harmonic mean of precision and recall: F1 = 2*(P*R)/(P+R)

Regression metrics

- Mean Intersection-over-Union
- Percentage of predictions with relative error on delta within 10%
- Mean Absolute Error (center and delta)
- Normalized Mean Absolute Error (delta)
  - MAE delta / mean delta duration
- Max Absolute Error (center and delta)

Combined metrics

- (Overall) Accuracy
  - Evaluate classification and regression simultaneously
  - Prediction is considered correct if (both)
    - Class prediction is correct
    - Bounding box min IoU threshold is satisfied (only if positive class)
  - # correct predictions / # of total predictions

Results (classification metrics)

	ResNet-8	ResNet-18	ResNet-34	MobileNet-v3
Accuracy	81%	83%	84%	77%
Weighted Avg Precision	83%	84%	84%	80%
Weighted Avg Recall	81%	83%	84%	77%
Weighted Avg F1-Score	82%	83%	84%	78%

Table 5. Classification metrics comparison

Results (regression metrics)

	ResNet-8	ResNet-18	ResNet-34	MobileNet-v3
Mean IoU	45.23%	46.51%	47.18%	48.10%
Perc. predicted delta within 10% relative error	26.77%	27.55%	21.46%	27.85%
MAE center	0.14s	0.14s	0.14s	0.14s
MAE delta	0.09s	0.09s	0.09s	0.10s
Normalized MAE delta	27.4%	26.84%	28.1%	29.44%
Max Absolute Error center	0.74s	0.69s	0.69s	0.77s
Max Absolute Error delta	0.73s	0.75s	0.74s	0.70s

Table 6. Regression metrics comparison

Results (combined metrics)

	ResNet-8	ResNet-18	ResNet-34	MobileNet-v3
Accuracy	58.67%	60.90%	64%	57.62%

Table 7. Combined metrics comparison

Ex: classification report ResNet-18

Class	Precision	Recall	F1-Score	Support
Breath	0.67	0.72	0.70	732
Laughter	0.68	0.85	0.76	579
Music	0.79	0.95	0.86	822
Nonfiller	0.97	0.88	0.92	6172
Uh	0.80	0.76	0.78	2598
Um	0.87	0.88	0.87	2446
Words	0.64	0.73	0.68	2292
Accuracy	0.83			
Macro Avg	0.77	0.82	0.79	15641
Weighted Avg	0.84	0.83	0.83	15641

Table 3. Classification Report using ResNet-18

Ex: confusion matrix ResNet-18

A / P	Breath	Laughter	Music	Nonfiller	Uh	Um	Words
Breath	529	73	5	0	46	43	36
Laughter	25	495	2	5	5	3	44
Music	4	16	778	1	3	0	20
Nonfiller	14	17	153	5401	101	36	450
Uh	66	28	9	25	1972	185	313
Um	33	9	8	6	142	2155	93
Words	119	86	35	123	199	65	1665

Table 4. Confusion matrix using ResNet-18. Actual class is on row; predicted class is on column

Loss trend ResNet-18 and MobileNet-v3

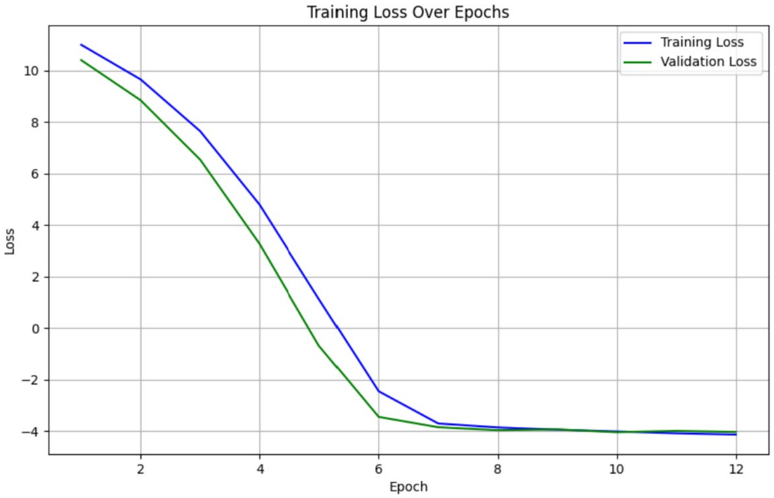

Figure 2. Training and validation loss over epochs with ResNet-18

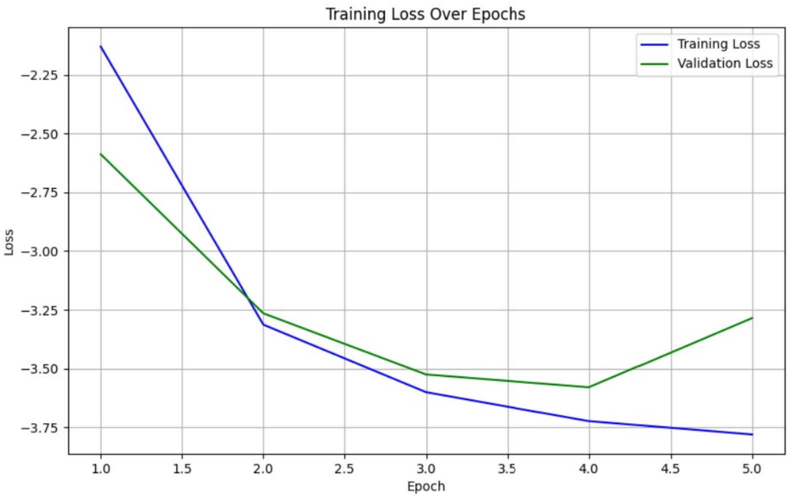

Figure 4. Training and validation loss over epochs with MobileNet-v3

Absolute Errors distributions ResNet-18 and MobileNet-v3

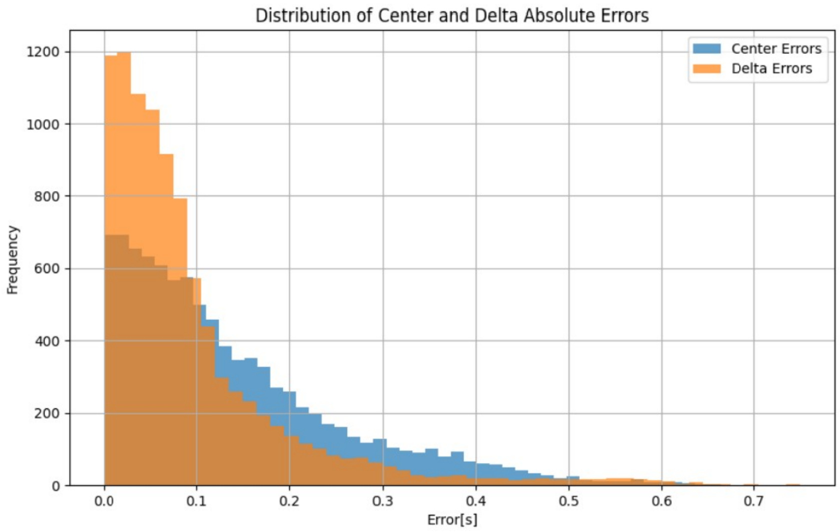

Figure 3. Distribution of absolute errors for center and delta with ResNet-18

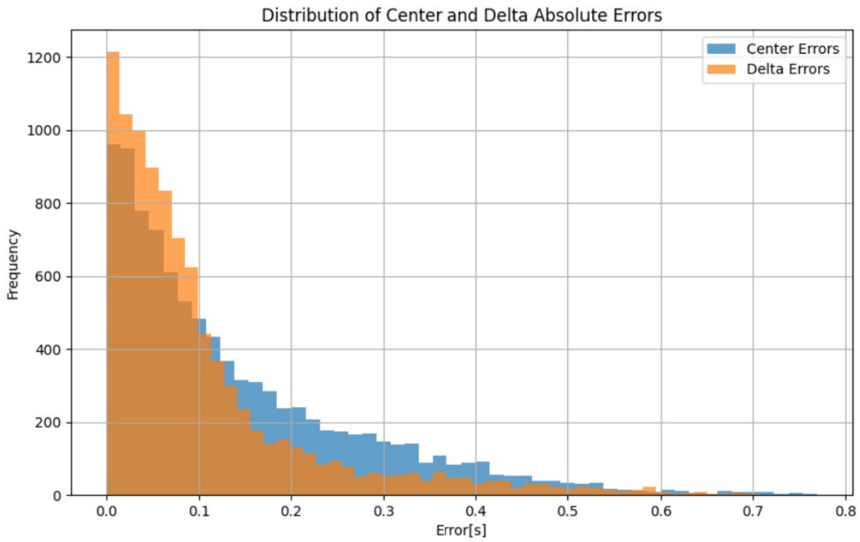

Figure 5. Distribution of absolute errors for center and delta with MobileNet-v3

Observations about results

- Different models, with different complexity performed similarly, both for classification and regression
  - Upper limit to performance is not considerably affected by model complexity
  - Performance limits may be due to
    - Architecture type (CNN)
    - Information loss during feature extraction process (power spectrogram instead of raw waveform)
    - Data quality
    - Imprecision in labeling process
      - Original dataset labels were produced through crowdsourcing
      - Confidence score for each label
      - Sometimes difficult to guess right class even for humans

Which model do we choose?

- All models showed similar results
  - Larger, more complex models only performed marginally better with respect to simpler ones
- There is no “best model” in absolute terms, among the ones tested
  - The choice depends on the particular computer architecture and available computational capabilities

Models and use cases

- Which model to choose depends on the particular context and computer architecture
  - *MobileNet-v3*: best choice for mobile CPU-only devices (e.g. smartphones, tablets, with RISC processors)
  - *ResNet-8*: best choice for desktop CPU-only devices (e.g. laptops). Good performance and quality
  - *ResNet-18*, *ResNet-34*: best choices for devices with GPU, since they provide the best performance and are designed to fully leverage the GPU parallelization capabilities

Other possible models

- *VGG-19*
  - Was not chosen because of too high number of trainable parameters for our computational means
    - Complex training
  - Requires high number of training samples
    - May cause overfitting with smaller datasets
- *MobileNet-v2*
  - Despite having very few parameters, operations are not optimized for GPUs
    - No leveraging of parallelization capabilities, even if GPU is available
    - Slow training

Qualitative observations

- Some classes are more difficult to recognize for the model
  - Filler words are typically better detected by the network
    - E.g. *Uh*, *Um*
    - “**Structured**” audio, shows common features independently of the speaker
  - Filler sounds are more difficult for the model to detect
    - E.g. *Music*, *Breath*, *Laughter*
    - “**Unstructured**” audio, high features variability, depends on particular speaker and context

Qualitative observations

- Some “regular” words (or a part of them) may be wrongly classified as filler ones
  - Some sounds or syllables may be interpreted as filler words themselves
  - E.g. *Um*brella (‘Um’ detected)
  - No direct way to address this problem
    - Architectural limit: solving this limitation would require using more complex models (e.g. based on transformers)
    - Out of scope for our purposes

Performance comparison to other models

- Comparing the performance of Speech-YALE to the results obtained using other models is difficult
  - There are not so many works which try to address the same task
    - More likely works which address "regular" words detection problem
  - They may use different data
    - Difficult to compare models trained on different datasets
  - They may use different metrics
    - E.g. model described in "Podcast Fillers" paper only performs classification on clips, so regression metrics are not available

Using trained model for inference

- We want to test the model qualitatively
- Use trained model to “clean” an audio signal
  - Remove silent intervals
  - Remove Filler Words
- Optional debug capabilities
  - Show start and end timestamps for each event occurrence
  - Extract each found occurrence and save it as a clip. Clips are grouped by event class (one folder per class)

Inference process

1) **Silence removal**: pre-process the signal removing silent intervals (power below a certain threshold)
2) **Split in 1s clips**: divide the signal in clips and form appropriate size batches to send as input to the trained model
3) **Post-processing**: use the predictions to produce the “clean” version of the audio signal, removing fillers and applying sinusoidal fade to make transitions smoother
4) **Output file saving**: save the clean signal on disk as an audio file

Alternative approach: regression vs classification

- Regression can be a difficult task
- **Idea**: approximate regression with multiple binary classification on discrete intervals
  - Instead of predicting continuous bounding box coordinates, split the [0, 1] range in 10 intervals and assign to each of them
    - 1 if that interval belongs to the bounding box
    - 0 otherwise

Alternative approach: regression vs classification

- However, results are inferior to those obtained by properly training an actual regressor
- Two main drawbacks
  - Not possible to predict bounding boxes which exceed the clip duration or boundary
    - But filler words are not usually entirely contained in one single clip
    - Requires more sophisticated inference algorithms with fractional stride
  - Classification only approximates regression
    - Lower accuracy in predicting bounding box coordinates