Voice activity detection for low-resource settings

Abhipray Sahoo†

CS230 Winter 2020 Stanford University   Video: https://youtu.be/dyGonCXYx8c

Motivation

Voice activity detection (VAD) is the task of detecting the presence of human speech in an audio signal containing speech and noise.

Applications: Front-end for speech recognition, speech enhancement and switching coding schemes in audio codecs.

Modern applications such as smart speakers require fast and efficient VAD. We use a deep learning approach for VAD constraining the memory footprint to work in low resource settings.

Dataset & Features

- **VCTK [1] dataset**: 109 speakers for 44 hours of speech.
- **Noisex-92**: babble, car, factory and white noises
- We augment VCTK by generating noisy versions of the original speech at 0dB and 10dB SNR.
- Mel-spectrogram features are extracted with 40 channels, 32ms frames and 16ms overlap.
- Ground truth labels are generated using per-frame energy-thresholding
- Train-dev-test split is 85%-10%-5%

Optimizing & Satisficing metrics

We care about both high true positive rate (TPR) as well as low false positive rate (FPR) i.e high precision and high recall. We use the harmonic mean of precision and recall, **F1 score**, as the optimizing metric. Satisficing metric is memory footprint of 10KB.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1} = 2\frac{\text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$$

Baseline: WebRTC

- WebRTC project [2] implements a VAD popular in open source projects
- Gaussian Mixture Model based
- F1 score for non-noisy speech is 0.819

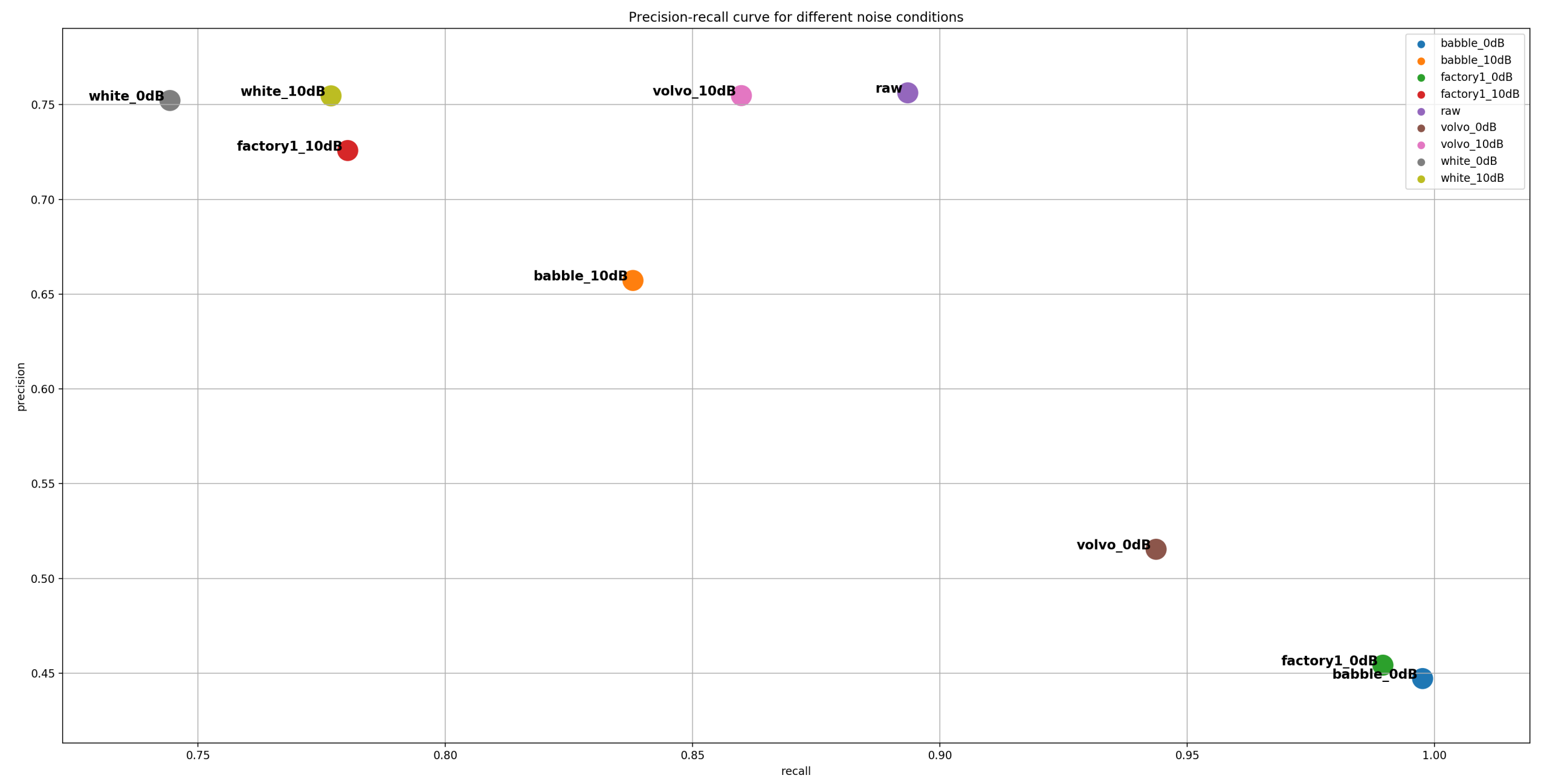
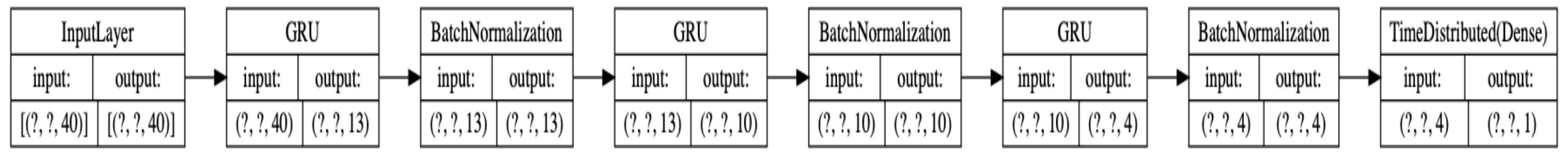


Fig. 1: Precision-recall curve for different noise conditions

GRU RNN Model

- Initial architecture choice: upto three GRU layers and one dense node with sigmoidal activation emitting probability of speech at each timestep. Batch Normalization and dropout layers are added in between layers. Optimize to minimize per-frame binary cross entropy loss.
- Bayesian hyperparameter tuning of dropout, learning rate, number of layers and number of nodes in each GRU layer.
- Three stages of training– large, medium and small model exploration; 90 trials total.



InputLayer — input: [(?, ?, 40)] — output: [(?, ?, 40)]

GRU — input: (?, ?, 40) — output: (?, ?, 13)

BatchNormalization — input: (?, ?, 13) — output: (?, ?, 13)

GRU — input: (?, ?, 13) — output: (?, ?, 10)

BatchNormalization — input: (?, ?, 10) — output: (?, ?, 10)

GRU — input: (?, ?, 10) — output: (?, ?, 4)

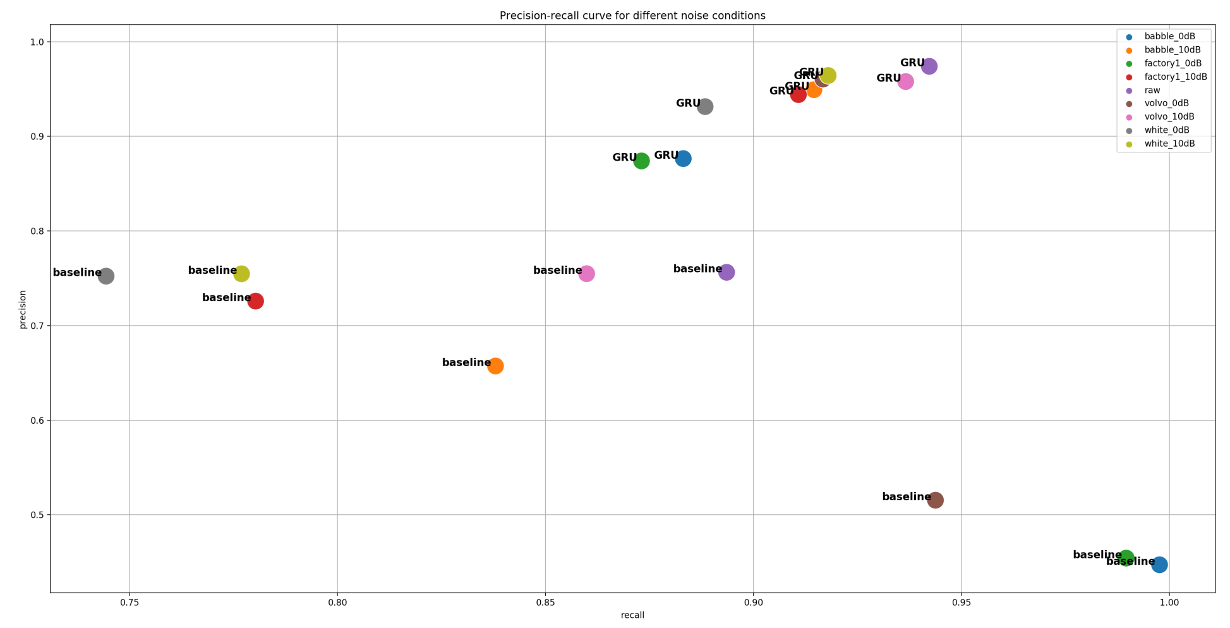
BatchNormalization — input: (?, ?, 4) — output: (?, ?, 4)

TimeDistributed(Dense) — input: (?, ?, 4) — output: (?, ?, 1)

Fig. 2: Best performing small model

Results

GRU outperfoms WebRTC in every noise condition. In non-noisy conditions, increased F1 score from 0.82 to 0.96. Final model has 3200 parameters; with 16-bit quantized floating point, it takes 6.4KB of memory.



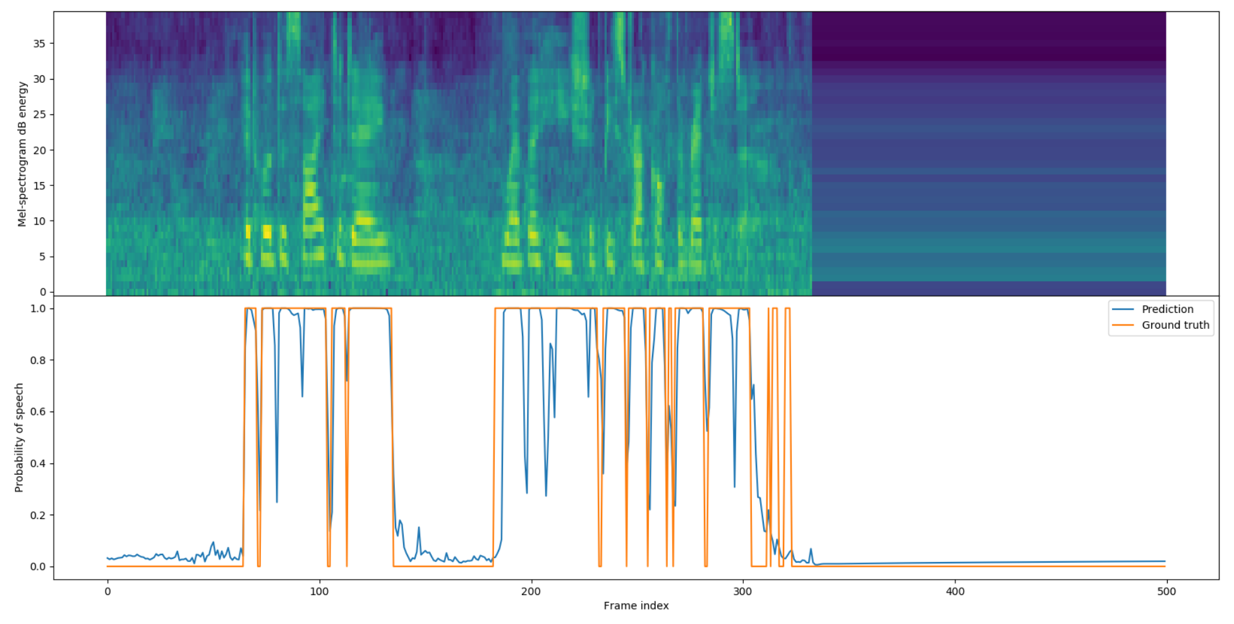


Fig. 4: Example prediction

Future work

- Re-train network with domain-adversarial training for deployment in different target acoustical domains.
- Quantize network weights to 8-bits to reduce memory footprint and increase computational speed.

References

[1] *CSTR VCTK Corpus*. https://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html. (Accessed on 01/19/2020).

[2] *webrtc/common_audio/vad - external/webrtc - Git at Google*. https://chromium.googlesource.com/external/webrtc/+/branch-