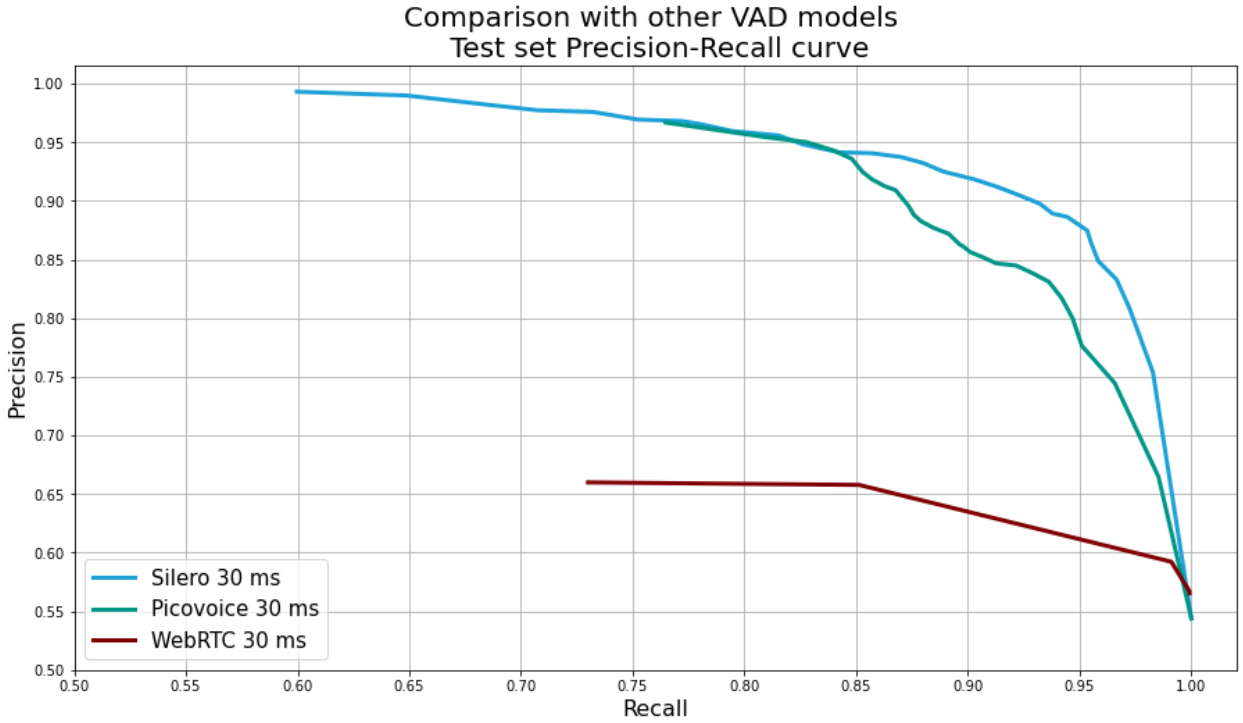
Обзор популярных и State-of-the-art методов

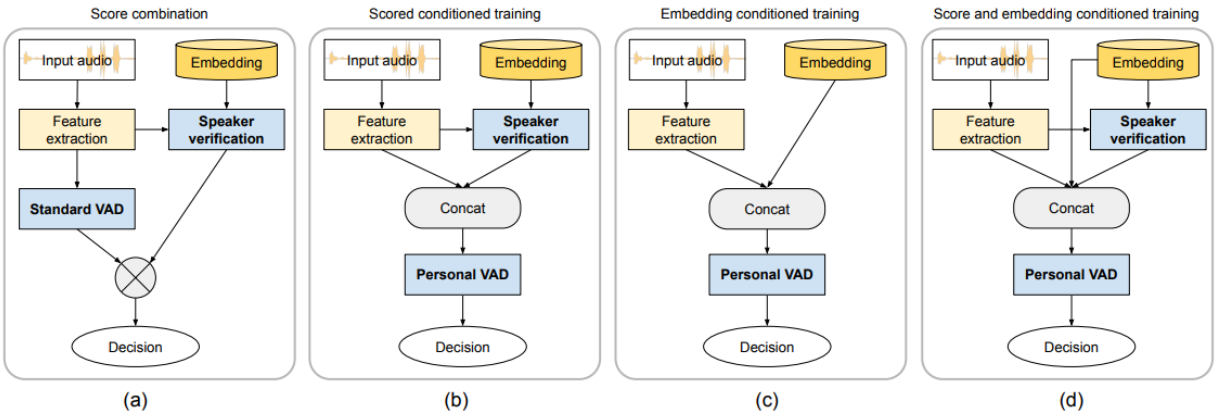
1. **Фреймворки с предобученными моделями**.
    a. WebRTC VAD (https://github.com/wiseman/py-webrtcvad). Фреймворк от Google. Один из самых популярных методов за счет свободного доступа, но достаточно нестабильный и неточный (хотя по заверениям разрабов, reportedly one of the best available, being fast, modern and free). Работает в трех режимах чувствительности, а так же на трех разных длинах фреймов – 10мс, 20мс и 30мс, что позволять использовать его в рамках техзадания как бейзлайн и как разметку
    b. Silero VAD (https://github.com/snakers4/silero-vad). Библиотека от Silero – команды, занимающейся обработкой речи. Очень сильно побивает WebRTC по качеству, но, к сожалению, работает только на фреймах 30+ мс.
    c. Picovoice's Cobra (https://github.com/Picovoice/cobra). Фреймворк для работы on-device, но есть и реализация на питоне. Также заметно бьет webrtc, немного уступая Silero. Из минусов – платный для коммерческого использования, инициализация происходит через веб-консоль.



2. Нейросетевые архитектуры.

Personal VAD: Speaker-Conditioned Voice Activity Detection (Google 2020) https://arxiv.org/pdf/1908.04284.pdf

ВАД, использующий эмбеддинги (d-вектора) целевого спикера для более точного и устойчивого выделения речи. Работает в реал-тайме по фреймам. Эмбеддинги генерируются маленькой трехслойной ЛСТМкой. Существует четыре вариации: Score combination (SC) (комбинация speaker verification и обычного ВАДа), Score conditioned training (ST) (модель верификации генерирует скор схожести, склеивает его с акустическими фичами, вад тренируется на на эту склейку), Embedding conditioned training (ET) (эмбеддинг склеивается с акустическими фичами, вад тренируется на склейку), и Score and embedding conditioned training (SET) (склейка скора и эмбеддинга с фичами)



Можно тренировать на кроссэнтропийный лосс:

$$L_{\mathrm{CE}}(y, \mathbf{z}) = -\log \frac{\exp(z^y)}{\sum_k \exp(z^k)},$$

Но тогда теряется разница между видами ошибок (шум-таргет, таргет-нетаргет, шум-нетаргет). Поэтому используется взвешенный попарный лосс:

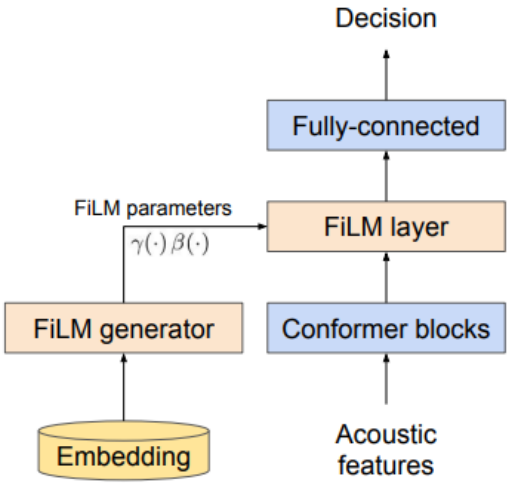
$$L_{\mathrm{WPL}}(y, \mathbf{z}) = -\mathbb{E}_{k \neq y}\left[w_{<k,y>} \cdot \log \frac{\exp(z^y)}{\exp(z^y) + \exp(z^k)}\right], \tag{10}$$

Омега(k,y) – вес между классами k и y. Вес «шум-нетаргет» меньше, чем «таргет-нетаргет» и «таргет-шум».

PVAD 2.0 (Google 2022) https://arxiv.org/pdf/2204.03793.pdf

Эмбеддинг и фичи – совсем разные сущности, которые к тому же добываются разными методами. Апдейт ПВАДа решает эту проблему двумя способами:

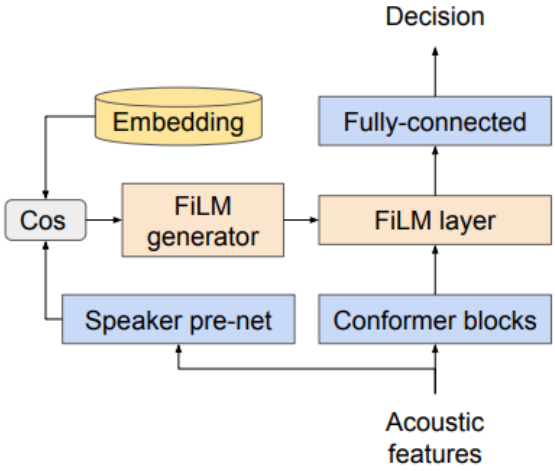
1. Слой FiLM. Аффинное преобразование, обобщающее скалирование, конкатенацию, смещение, что более репрезентативно, чем использование этих операций по отдельности



FiLM-генератор принимает на вход эмбеддинг и генерирует векторы сдвига и скейла, с теми же размерностями, что и вход слоя. Затем слой преобразует вход:

$$\mathrm{FiLM}(\mathbf{h}) = \gamma(\mathbf{e}^{\mathrm{target}}) \cdot \mathbf{h} + \beta(\mathbf{e}^{\mathrm{target}})$$

2. Speaker embedding modulation



Пре-нет принимает в себя фичи, делает из них эмбеддинг, дальше считается косинусный скор между настоящим эмбеддингом и пре-нетовским, и этим скором модулируется выход конформера

$$\mathbf{e}^{\mathrm{prenet}} = \mathrm{PreNet}(\mathbf{x})$$

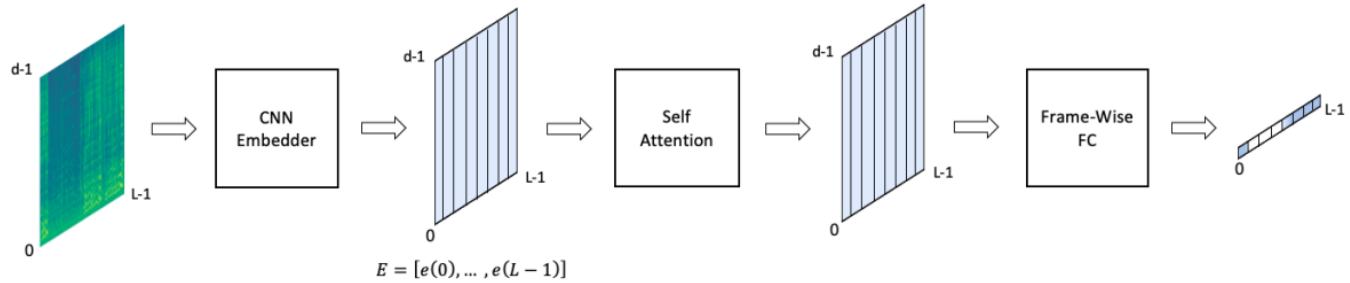
$$\mathbf{s} = \cos(\mathbf{e}^{\mathrm{prenet}}, \mathbf{e}^{\mathrm{target}})$$

$$\mathrm{FiLM}(\mathbf{h}) = \gamma(\mathbf{s}) \cdot \mathbf{x} + \beta(\mathbf{s})$$

Может работать без энроллмента; вектор эмбеддинга зануляется, ground truth лейблы нон-таргета заменяются на таргет. Работает со стримингом (конформер), модель квантизована в инт8

CNN self-attention voice activity detector (OriginAI, 2022) https://arxiv.org/pdf/2203.02944.pdf

ВАД из сверточного эмбеддера и self-attention энкодера. Сверточная сетка добывает зависимости между фреймами, энкодер берет эмбеддер и ищет похожие фичи во фреймах.



На вход принимается мел-спектр, из него генерится эмбеддинг той же размерности, который кидается в multi-headed attention,

$$\text{MultiHead} = Concat(head_0, \ldots, head_{H-1})W^O$$

where

$$head_i = softmax(\frac{Q_i K_i^T}{\sqrt{d}})V_i,$$
$$Q_i = E \cdot W_i^Q, \;\; K_i = E \cdot W_i^K, \;\; V = E \cdot W_i^V$$

из которого считается усредненный аттеншн каждого фрейма к каждому

$$\text{AverageAttention} = \frac{1}{H}\sum_{i=0}^{H-1} softmax(\frac{Q_i K_i^T}{\sqrt{d}}).$$

И полируется нормализацией и фулл-коннектом

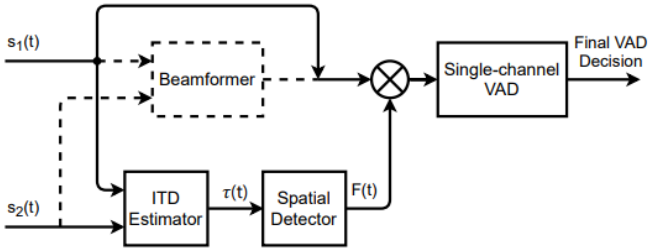
Эмбеддер состоит из 4 сверточных слоев с батч-нормом, PReLU и макс-пулингом

Выход сверток – [L, F’ , C], этот выход сплющивается в [L, F’ ·C], затем фуллконнект добивает это до эмбеддинга размером d.

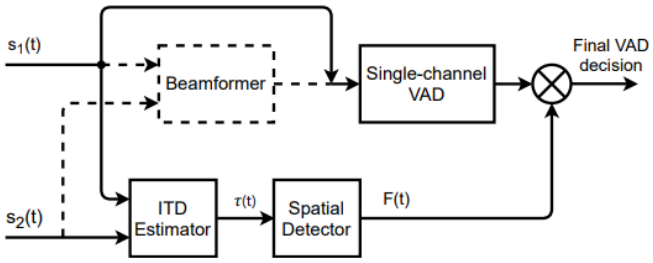
Improvement of Noise-Robust Single-Channel Voice Activity Detection with Spatial Pre-processing (2021)
https://arxiv.org/pdf/2104.05481.pdf

Шумоустойчивый вад, использующий пространственный детектор и бимформинг

$$s_1(k) = a_1(k) \cdot x(k) + n_1(k),$$
$$s_2(k) = a_2(k) \cdot x(k+\tau) + n_2(k),$$



(a) *Spatial filter method with an optional beamformer (dashed line)*



(b) *Spatial VAD method with an optional beamformer (dashed line)*

Figure 1: *Flowchart of the proposed method for the two combination approaches*

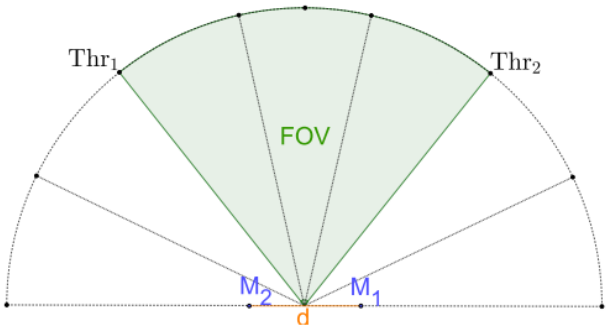


Figure 2: *Overview of the dual microphone array, thresholds and FOV*

Пространственный фильтр определяет, находится ли целевое направление в пределах «поля зрения» микрофонов в текущий момент времени (GCC-PHAT). Фильтр может быть применен до вада, либо скомбинирован с решением вада. Кроме того, можно использовать бимформер для бОльшего улучшения качества результатов

В качества непосредственно ВАДа можно использовать любую подходящую модель

A HYBRID CNN-BILSTM VOICE ACTIVITY DETECTOR (2021) https://arxiv.org/pdf/2103.03529.pdf

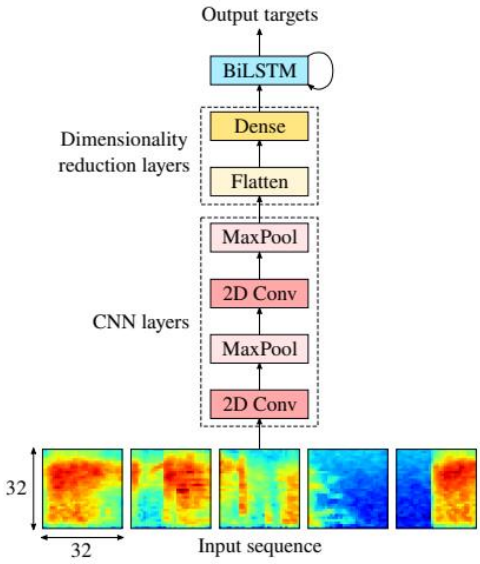


Fig. 1. Block diagram of the CNN-BiLSTM VAD.

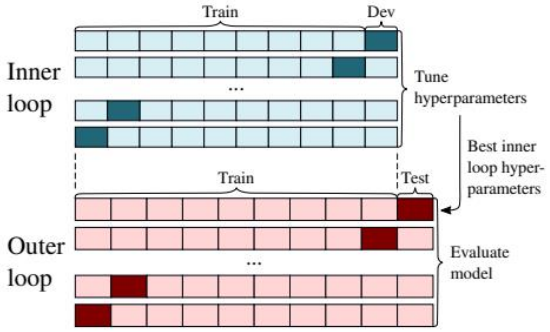
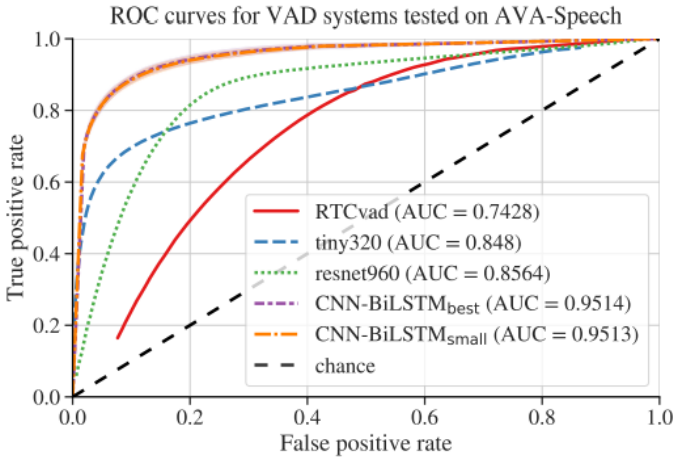


Fig. 2. Nested k-fold cross-validation procedure for hyperparameter tuning and model evaluation.



Тяжелая нереалтаймовая модель, но с неплохими метриками качества. Работает по мел-спектру

Свертки с пулингом и flatten понижают размерности для билстм слоев. В целом, в сетке ничего особенного, но она достаточно сильно выигрывает у state-of-the-art моделей по ROC

Выбранная архитектура

DenseNet (2018) https://arxiv.org/pdf/1608.06993.pdf

Так как задание подразумевает сократить задержку до минимума, то возникает очевидное желание отказаться от LSTM архитектур в пользу сверточных решений. Одним из таких проверенных решений является DenseNet. Как и многие другие архитектуры, изначально он был предназначен для изображений, но может быть легко адаптирован под вход из аудио-фичей.

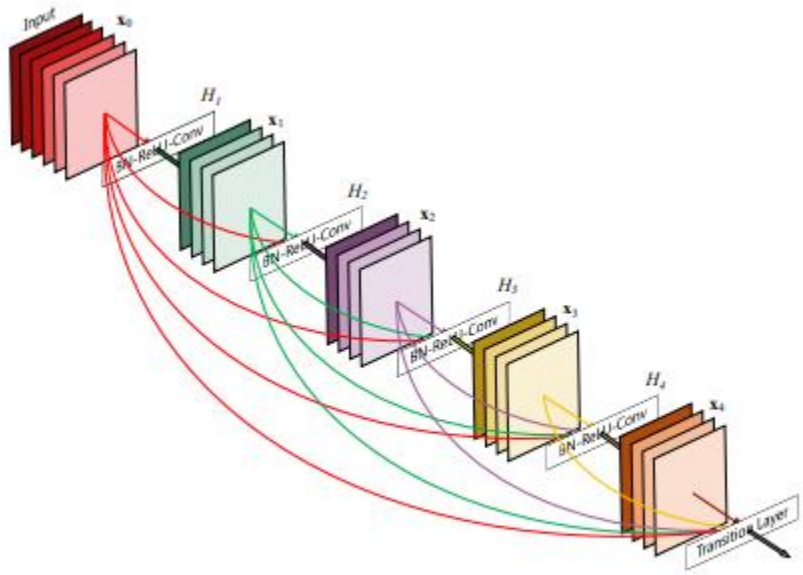


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

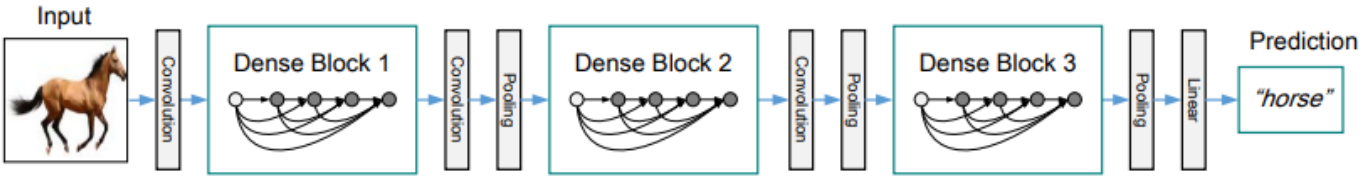


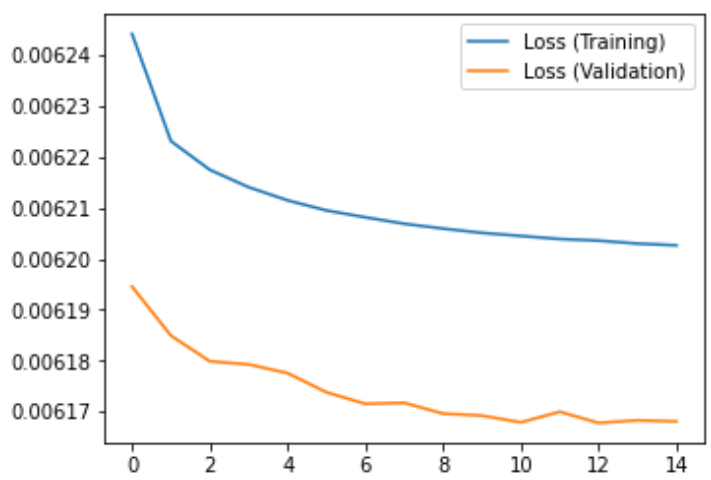
Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

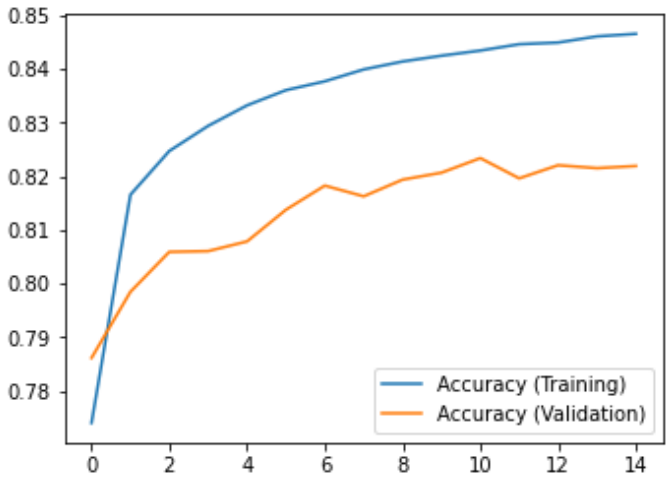
Идея densenet – residual можно пробрасывать не только через слой, а в любой последующий слой. Таким образом, каждый слой densenet прокидывает резидуал во все последующие слои. Важно отметить, что, в отличие от ResNet, фичи, прежде чем они будут переданы в следующий слой, не суммируются, а конкатенируются (channel-wise concatenation). При этом количество параметров сети DenseNet намного меньше, чем у сетей с такой же точностью работы. Авторы утверждают, что DenseNet работает особенно хорошо на малых наборах данных, что играет важную роль в условиях данного эксперимента (об этом подробнее в ноутбуке).

Дополнительные ссылки на использованные статьи и репозитории также указаны в ноутбуке.

В качестве метрики качества используется accuracy, посколько мы решаем задачу бинарной классификации.

Графики лосса и accuracy на тренировке и валидации:

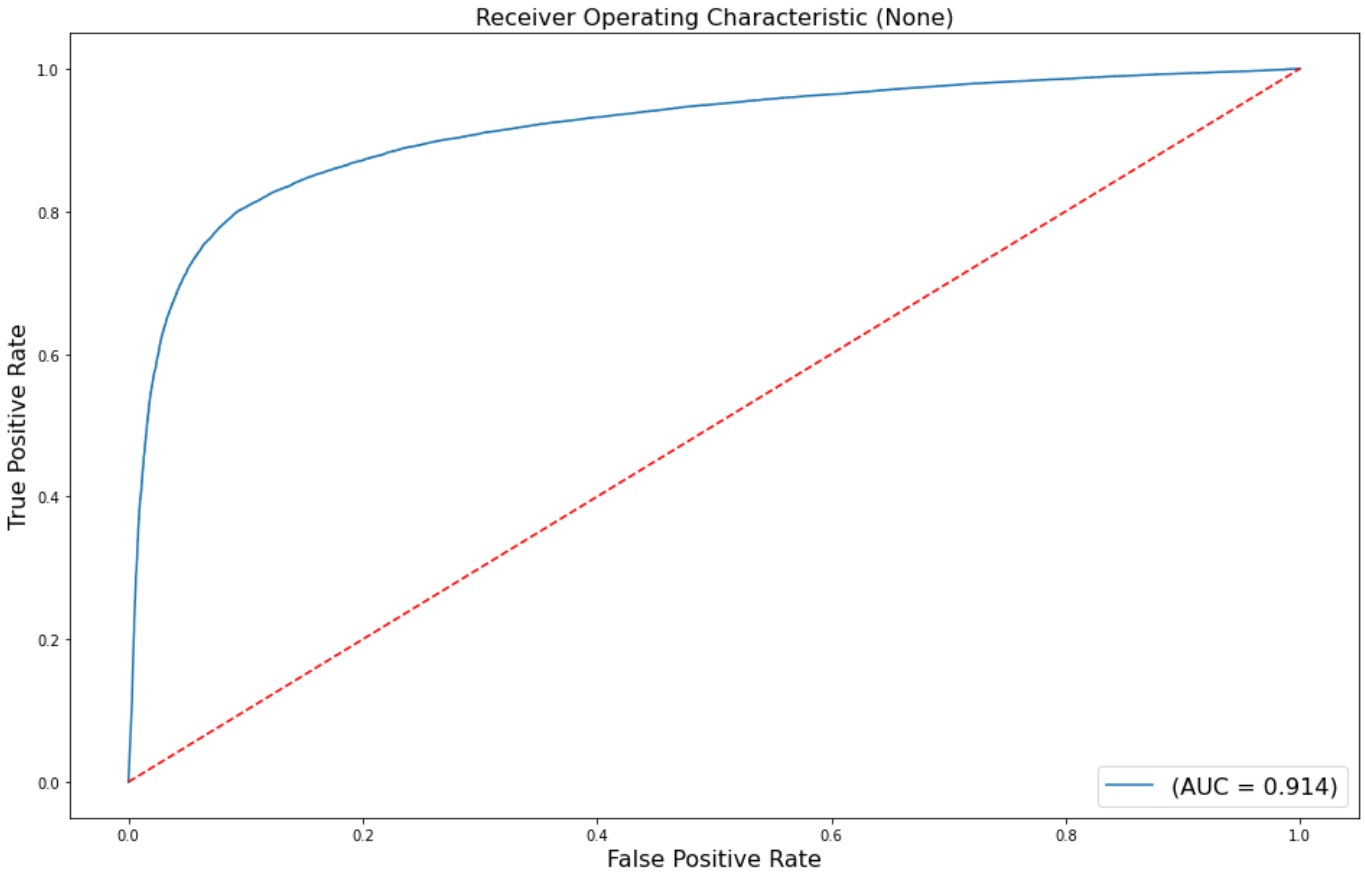


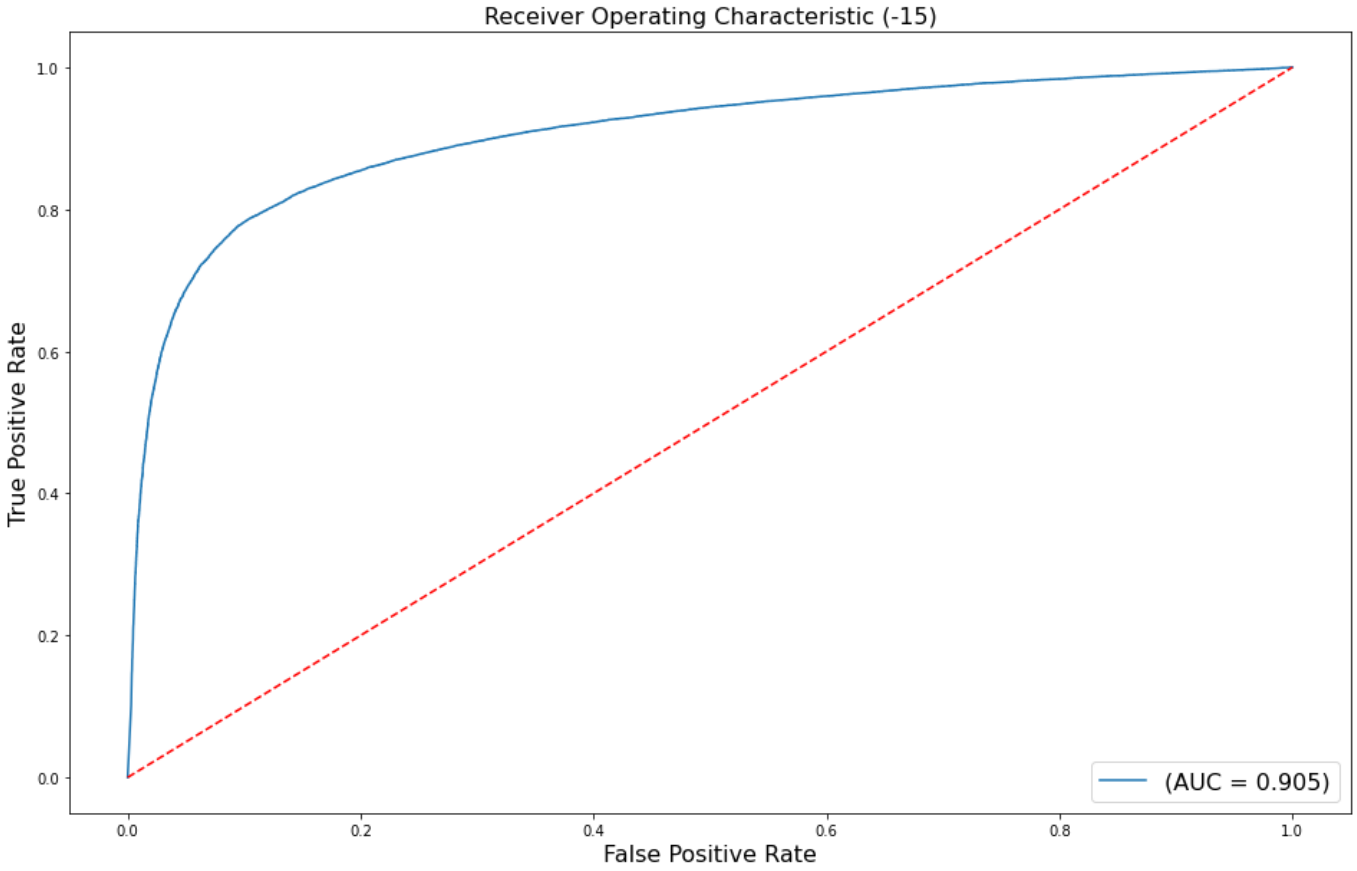


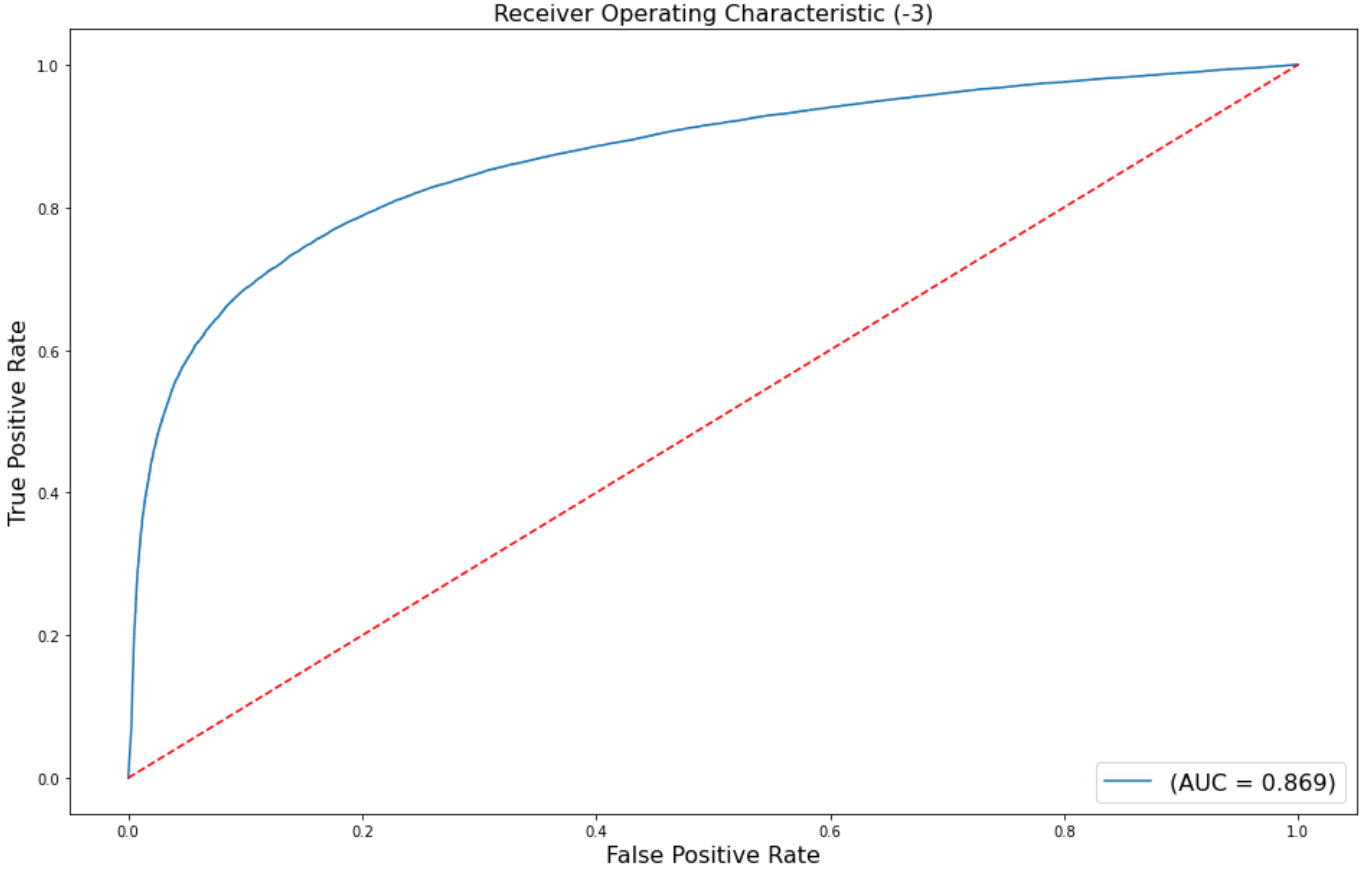
Accuracy достаточно сильно просаживается. Чтобы этого избежать, возможно нужно:

1) Увеличить время обучения
2) Увеличить количество параметров сети
3) Использовать другие, более большие и разнообразные датасеты (train-clean-, train-other- для речи, MUSAN для шумов)

ROC-кривые на трех уровнях шума




Receiver Operating Characteristic (-15)
True Positive Rate
False Positive Rate
(AUC = 0.905)
0.0
0.2
0.4
0.6
0.8
1.0


Receiver Operating Characteristic (-3)
True Positive Rate
False Positive Rate
(AUC = 0.869)
0.0
0.2
0.4
0.6
0.8
1.0

Пороги:

Без шума:

FAR: 28.88% for fixed FRR at 9.43%
FRR: 60.36% for fixed FAR at 1.02%
EER: 15.16%

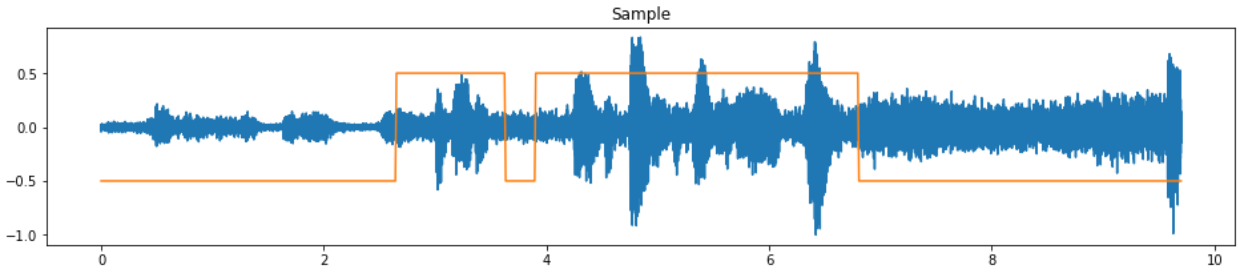
Слабый шум:

FAR: 41.86% for fixed FRR at 7.30%
FRR: 61.73% for fixed FAR at 1.03%
EER: 16.41%

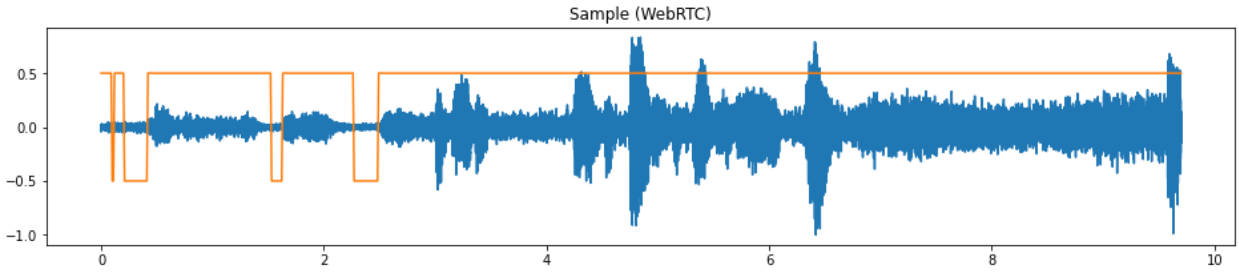
Сильный шум:

FAR: 91.00% for fixed FRR at 1.03%
FRR: 68.07% for fixed FAR at 1.01%
EER: 20.64%

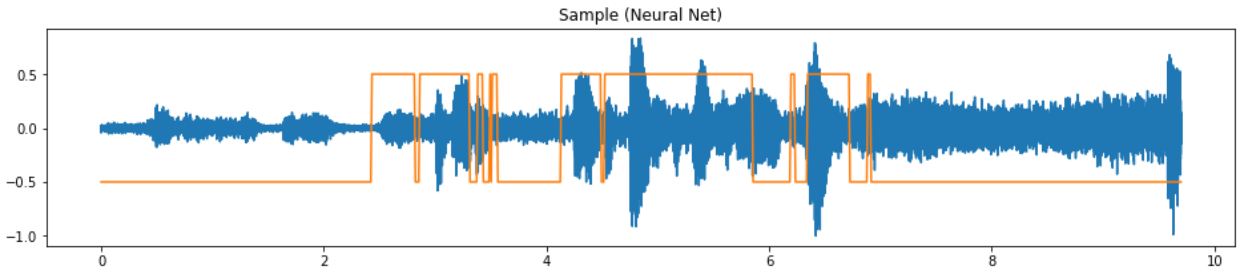
Примеры на семпле из тестового датасета:



ground truth



WebRTC



DenseNet

С учетом достаточно маленького размера модели, а так же достаточно ограниченных тренировочных данных, модель достигла достаточно неплохих результатов, которые потенциально могут быть улучшены увеличением датасета и количества параметров сети

Предложенный тестовый датасет обработан полученной моделью, а также с помощью WebRTCVAD.

Результаты обработки находятся в файлах for_devs.txt и for_devs_webrtc.txt. Также в https://drive.google.com/drive/folders/17MvT3Feip8QIX3x_5ce8_AUsQ7TmHkMe?usp=sharing лежит файл markdown_output.html с графическими результатами обработки тестового датасета.

Из-за особенностей обработки файлов моделью, последние 30 фреймов каждого файла будут нулями.

Так как разметка с помощью WebRTCVAD не может являться идеальным ground truth, то нет смысла подсчитывать количественные метрики. Визуализация из markdown_output.ipynb позволяет оценить общее качество модели: она плохо справляется с реверберированными файлами (в связи с отсутствием аугментации импульсными откликами), но качественнее делит речь на сегменты (распознает паузы между словами там, где webrtc предсказываем длинный лейбл из единичек). Также полученная модель лучше справляется с диффузными шумами (шум улицы или дождь), поскольку датасет шумов состоял преимущественно из них.

Код, дополнительные комментарии, ссылки на статьи и репозитории находятся в DenseNet_Vad.ipynb