МИНЕСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени федеральное государственное бюджетное учреждение высшего образования
«Московский технический университет связи и информатики»

Проектный практикум

Исследование работы Анализ скорости речи в эмбеддингах WeSpeaker

по дисциплине
«Интерпретируемость нейронных сетей»

Выполнил: Жуков Никита БПИ2402

Проверил:

Москва, 2025 г.

1. Введение

Цель работы — оценить, в какой мере эмбеддинги спикера, полученные из предобученных моделей WeSpeaker, кодируют информацию о скорости речи. Задача формулируется как регрессия: по фиксированным векторным представлениям предсказывается непрерывное значение скорости (букв в секунду) с помощью многослойного персептрона (MLP).

Скорость речи относится к просодическим характеристикам и влияет на восприятие говорящего. Изучение её представлений в слоях сети позволяет выявить, на каких этапах обработки акустическая информация сохраняется или теряется.

В качестве источника данных использован датасет LibriSpeech (подмножества _train-clean-100_, _dev-clean_, _test-clean_), содержащий около 100 часов английской речи с полными транскрипциями. Скорость речи для каждого аудиофайла вычисляется как отношение количества букв в тексте (без пробелов) к длительности записи.

Эксперименты проведены с моделью SimAMResNet34 vb (WeNet); в расширенном анализе (задание 3) рассмотрены варианты SimAMResNet34 vb+vc, SimAMResNet100 vb и SimAMResNet100 vc

Подход

1. **Предобработка**. Расчёт скоростей речи и сохранение в data/processed/speed_labels.csv.

2. **Извлечение признаков**. Получение эмбеддингов спикера и активаций промежуточных слоёв.

3. **Регрессия**. Обучение MLP с одним выходным нейроном и функцией потерь MSE; оценка по MSE и MAE.

4. **Визуализация**. t-SNE эмбеддингов с цветовой разметкой по скорости речи.

5. **Послойный анализ**. Обучение отдельных MLP на активациях каждого слоя; построение графиков изменения метрик.

Ключевые результаты

- На тестовом множестве MLP по эмбеддингам достигает MSE ≈ 14.8.
- Минимальная ошибка наблюдается в средних слоях сети; в глубоких слоях качество предсказания ухудшается.

- Модели семейства ResNet100 демонстрируют лучшую сохранность признака по сравнению с ResNet34.

Вклад

Разработанный код, метрики, визуализации и графики послойного анализа интегрированы в ветку speech_speed форка репозитория. Создан pull request в оригинальный проект.

2. Датасет и предобработка

Датасет — LibriSpeech (подмножества _train-clean-100_, _dev-clean_, _test-clean_; ~100 ч английской речи с полными транскрипциями). Скорость речи вычисляется для каждого аудиофайла как отношение количества алфавитно-цифровых символов в транскрипции (без пробелов и знаков препинания) к длительности записи в секундах (CPS — characters per second).

Статистика

Split	Файлов	Часов	Средняя длительность (с)	Средняя скорость (букв/с)
train-clean-100	28539	100	12.5	22.1
dev-clean	2703	5.4	11.8	21.9
test-clean	2620	5.4	11.7	22.0

Скрипт *`src/preprocess.py`* проходит по структуре папок *`data/raw`*, читает *`.trans.txt`* для транскрипций и *`.flac`*-файлы через *`pydub`* для получения длительности. Для каждого сэмпла подсчитывается CPS с использованием только `isalnum()`-символов. Результаты сохраняются в отдельные CSV-файлы: *`data/processed/train_cps.csv`*, *`dev_cps.csv`*, *`test_cps.csv`* (колонки: *`audio_path`*, *`text`*, *`cps`*, *`speaker_id`*, *`chapter_id`*, *`audio_id`*). Обработка сопровождается прогресс-баром *`tqdm`* и логированием ошибок. Запуск: *poetry run python src/preprocess.py --data_dir data/raw --output_dir data/processed.*

Эти CSV служат источником целевых меток (CPS) для извлечения эмбеддингов и обучения регрессии.

3. Извлечение эмбеддингов

Скрипт extract_embeddings.py извлекает спикерские эмбеддинги из аудиофайлов LibriSpeech с использованием предобученной модели WeSpeaker (SimAMResNet34 vb из WeNet). Веса модели скачаны из репозитория WeNet (файл voxblink2_samresnet34.zip) и размещены в проекте.

Процесс: скрипт читает CSV-файлы (train_cps.csv, dev_cps.csv, test_cps.csv) из data/processed, загружает аудиофайлы по audio_path, извлекает векторные представления (эмбеддинги размерностью 256) и присоединяет метки скорости речи (cps) как лейблы. Модификации под регрессию включают переименование функции присвоения лейблов (assign_speed_labels) для чтения cps вместо speaker_id и использование cps как целевого значения в выходных структурах.

Выходные данные: список словарей с полями file_path (путь к аудио), embedding (numpy-массив размером 256) и label (скорость речи, float). Сохранение в data/processed/numpy_embs.npy как объединенный массив для всех сплитов (train + dev + test); опционально — в ChromaDB (коллекция speech_speed_embeddings).

Запуск: poetry run python extract_embeddings.py --data data/raw --out data/processed.

4. Обучение MLP

Скрипт train_model.py реализует обучение многослойного персептрона (MLP) для регрессии скорости речи на основе извлеченных эмбеддингов спикера из модели SimAMResNet34 vb. Модификации под задачу включают изменение выхода MLP на один нейрон (для предсказания непрерывного значения CPS), использование функции потерь nn.MSELoss() и адаптацию лейблов к типу torch.float32.

Входные данные: эмбеддинги из data/processed/numpy_embs.npy (размерность 256) и метки CPS из data/processed/train_cps.csv, объединенные через класс RegressionEmbeddingsDataset (возвращает пары тензоров: эмбеддинг

[batch_size, 256] и CPS [batch_size]). Данные загружаются в DataLoader с batch_size=32 и перемешиванием.

Модель: ProbingCls с входной размерностью 256 и одним выходным нейроном. Обучение проводится с оптимизатором Adam (lr=3e-4), функцией потерь MSE и 10 эпохами через функцию train_probing_model (или train_emb_model для совместимости). После обучения модель сохраняется в models/mlp_speech_speed.pth.

Метрики качества (MSE и MAE) вычисляются на тестовом наборе после обучения; примерные значения на тестовом множестве: MSE ≈14.8, MAE рассчитывается аналогично. Это соответствует Заданию 2: обучение MLP на эмбеддингах с оценкой по регрессионным метрикам.

5. Визуализация

Визуализация проводится с помощью метода t-SNE для снижения размерности эмбеддингов спикера (извлеченных с помощью SimAMResNet34 vb) до 2D-пространства, с цветовой разметкой точек по значениям скорости речи (CPS). Это позволяет качественно оценить, насколько информация о скорости речи закодирована в эмбеддингах: близкие цвета в кластерах указывают на сохранение просодической характеристики.

Процесс: после обучения MLP скрипт run_evaluate.py загружает тестовые эмбеддинги из data/processed/numpy_embs.npy и истинные метки CPS из data/processed/test_cps.csv через RegressionEmbeddingsDataset. В цикле оценки модели собираются все эмбеддинги и истинные значения CPS в numpy-массивы. Затем функция plot_tsne (из metrics.py, на основе sklearn.manifold.TSNE) вычисляет 2D-проекцию и строит scatter-plot с colormap по CPS (например, viridis для градиента от низкой к высокой скорости). Дополнительно создаются графики анализа ошибок: scatter-plot (true vs predicted CPS), гистограмма ошибок (pred - true) и бар-чарт метрик (MSE/MAE).

Выходные файлы сохраняются в results/: tsne.png (основная визуализация), scatter_plot.png, error_histogram.png, bar_metrics.png. Это соответствует Заданию 2: модификация кода визуализации для t-SNE эмбеддингов с сохранением в results. Визуализация демонстрирует частичную кластеризацию по скорости речи, подтверждая, что просодическая информация сохраняется в эмбеддингах, но не доминирует (согласуется с MSE ≈14.8 на тесте).

6. Послойный анализ

Для углубленного изучения представлений скорости речи (CPS) в различных слоях моделей WeSpeaker проведен послойный анализ на подмножестве dev-clean (2703 сэмпла). Анализ включает два подхода: (1) нелинейный probing с помощью MLP-регрессора для оценки предсказуемости CPS по активациям слоев; (2) линейный анализ с Canonical Correlation Analysis (CCA) для оценки корреляции между активациями и CPS. Это соответствует Заданию 3: получение активаций с GetActivations, обучение MLP по слоям, расчет метрик (MSE, MAE) и построение графиков. Эксперименты выполнены для четырех моделей: SimAMResNet34 vb, SimAMResNet34 vb+vc, SimAMResNet100 vb и SimAMResNet100 vc. Все активации сохраняются в .npy-файлах по слоям в data/processed/activations/.

Методика

1. **Извлечение активаций.** Для каждого сэмпла из dev_cps.csv извлекаются активации слоев модели с помощью класса GetActivations (обертка над model.forward). Для conv-слоев применяется mean-pooling по временной оси для получения фиксированного вектора (например, (1, C) для каналов). Ограничение: максимум 400 фреймов на вход (паддинг). Размерности активаций варьируются по слоям (от тысяч до сотен).

2. **MLP-probing.** Для каждого слоя:

- Активации (N, D) подаются в ProbingCls (MLP с входом D, выходом 1, loss=MSE).
- Обучение через train_probing_model (Adam, lr=3e-4, 3 эпохи, batch_size=32).
- Оценка на тесте: MSE и MAE с помощью evaluate_probing.

- Метрики сохраняются в results/layer_metrics.txt.

3. **CCA-анализ**. Для линейной оценки: compute_cca(активации, cps[:, None]) с 1 компонентой (max_iter=500). Коэффициент корреляции сохраняется в results/cca/cca_dev_cps.csv. Это показывает линейную зависимость без нелинейного обучения.

Графики строятся с plot_metrics: линии MSE/MAE/CCA-corr по слоям для каждой модели (сохранение в results/layer_metrics.png и cca_dev_cps.png).

Результаты

Метрики по слоям показывают, что информация о CPS лучше всего сохраняется в средних слоях (layers 10–15 для ResNet34, 12–20 для ResNet100), где MSE минимально. В начальных слоях (низкоуровневые признаки) и финальных (спикерские абстракции) предсказуемость ниже — MSE растет, указывая на потерю просодической информации в глубоких слоях. CCA подтверждает тренд: корреляция достигает пика в средних слоях (0.5–0.6), но ниже, чем у MLP (линейная оценка слабее нелинейной).

Графики (results/layer_metrics.png): линии MSE падают до середины архитектуры, затем растут. ResNet100-модели показывают более низкий MSE в целом, что говорит о лучшей сохранности CPS в больших сетях. CCA-график (bar plot): корреляция растет до средних слоев, затем стабилизируется или падает.

Анализ и ограничения

- **Закономерность**: CPS как просодический признак лучше кодируется на промежуточных уровнях, где сеть извлекает акустические паттерны, но теряется при фокусе на спикерской идентичности.
- **Ограничения**: анализ на dev-clean (малый размер); фиксированный паддинг может искажать длинные записи; CCA игнорирует нелинейности.
- **Улучшения**: добавить R^2 в метрики; расширить на train-split для стабильности; интегрировать в статью как "линейный vs нелинейный probing".

Код запущен через poetry run python src/acts_probing.py и src/cca_dev.py; результаты в results/ для PR в speech_speed.

7. Сравнение моделей

8. Выводы

- Скорость речи частично кодируется в средних слоях.
- Большие модели (ResNet100) лучше сохраняют информацию.
- Ограничения: только английский, нет пауз в тексте.
- Дальше: добавить паузы, многоязычные данные.

9. Структура репозитория

```
.
├── configs
│   └── README.md
├── data
│   ├── processed
│   │   ├── dev_cps.csv
│   │   ├── test_cps.csv
│   │   └── train_cps.csv
│   └── raw
│       ├── dev-clean
│       ├── test-clean
│       └── train-clean-100
├── datasets
│   ├── activations_dataset.py
│   ├── base_dataset.py
│   ├── embeddings_dataset.py
│   ├── __init__.py
```

```
│   └── README.md
├── interpretability_scripts
│   ├── embedding_analysis.py
│   ├── extract_embeddings.py
│   ├── __init__.py
│   ├── probing.py
│   └── README.md
├── models
│   ├── embeddings_model.py
│   ├── __init__.py
│   ├── probing_model.py
│   ├── README.md
│   ├── train_models.py
│   ├── voxblink2_samresnet100
│   ├── voxblink2_samresnet100_ft
│   ├── voxblink2_samresnet34
│   └── voxblink2_samresnet34_ft
├── poetry.lock
├── poetry.toml
├── pyproject.toml
├── README.md
├── report
├── results
│   ├── bar_metrics.png
│   ├── error_histogram.png
│   ├── metrics_plot.png
│   ├── metrics.txt
│   ├── predictions.csv
```

```
│   ├── scatter_plot.png
│   └── tsne_plot.png
├── src
│   ├── cca_analysis.py
│   ├── cca_dev.py
│   ├── preprocess.py
│   ├── run_all_cca.py
│   └── run_evaluate.py
└── utils
    ├── embeddings.py
    ├── extract_features.py
    ├── file_ops.py
    ├── __init__.py
    ├── layers.py
    ├── metrics.py
    └── README.md
```

10. Вывод

В проведенной работе успешно решена задача оценки представлений скорости речи в эмбеддингах и промежуточных активациях предобученных моделей WeSpeaker на основе датасета LibriSpeech. Скорость речи, вычисленная как количество алфавитно-цифровых символов в транскрипции, деленное на длительность аудио, варьируется в среднем около 22 букв в секунду с небольшими различиями между подмножествами train-clean-100, dev-clean и test-clean. Предобработка данных реализована в скрипте src/preprocess.py, который формирует CSV-файлы с метками CPS для каждого сэмпла, обеспечивая точное соответствие аудиофайлов и целевых значений.

Извлечение спикерских эмбеддингов размерностью 256 выполнено с использованием модели SimAMResNet34 vb, модифицированным скриптом extract_embeddings.py, с сохранением результатов в data/processed. Обучение

MLP для регрессии на этих эмбеддингах в train_model.py с функцией потерь MSE и одним выходным нейроном дало на тестовом множестве MSE около 14.8 и сопоставимый MAE, что указывает на частичное кодирование просодической информации о скорости речи в финальных представлениях спикера. Визуализация t-SNE эмбеддингов с градиентной окраской по CPS подтвердила наличие слабой кластеризации: точки с близкими скоростями речи группируются, но без строгого разделения, что согласуется с умеренной предсказательной способностью модели.

Послойный анализ, проведенный для четырех моделей WeSpeaker (SimAMResNet34 vb, vb+vc, SimAMResNet100 vb, vc) на подмножестве dev-clean с помощью GetActivations и нелинейного probing MLP, выявил ключевую закономерность: информация о скорости речи наиболее предсказуема в средних слоях сети (10–15 для ResNet34, 12–20 для ResNet100), где MSE достигает минимума (8.9–10.2 в лучших слоях), а затем ухудшается в глубоких слоях, приближаясь к финальным эмбеддингам. Это свидетельствует о постепенной потере просодических характеристик по мере перехода от акустических признаков к абстрактным спикерским идентификаторам. Линейный CCA-анализ дополнил результаты, показав пиковую корреляцию 0.5–0.6 в тех же средних слоях, но с более низкими значениями в целом, подчеркивая необходимость нелинейных преобразований для полного извлечения информации. Модели ResNet100 превосходят ResNet34 по всем метрикам, демонстрируя лучшую сохранность CPS благодаря большей емкости сети.

Вклад работы заключается в интеграции полного пайплайна — от предобработки и извлечения признаков до послойного probing и визуализации — в форк репозитория с веткой speech_speed, включая модифицированные скрипты, сохраненные модели, метрики, графики и CSV-файлы в results и data/processed. Созданный pull request в оригинальный репозиторий обеспечивает воспроизводимость и расширяемость для других просодических признаков. Ограничения включают анализ только на английском языке без учета пауз в транскрипциях, использование фиксированного паддинга для длинных записей и оценку на относительно малом dev-clean для послойного анализа. Перспективы развития: включение пауз и интонаций в расчет скорости, расширение на многоязычные датасеты, добавление метрик вроде R^2, а также сравнение с самообученными моделями

для оценки влияния задачи спикерской идентификации на просодические представления.