Пошаговый план экспериментов для MVP-детектора аудио-дипфейков

Ниже приведён холодный пошаговый план простых экспериментов для создания минимального жизнеспособного продукта (MVP) по распознаванию аудио-дипфейков в среде **GitHub Codespaces**. План базируется на актуальных исследованиях и включает готовые фрагменты кода. Все шаги рассчитаны на один рабочий день.

1. Подготовка среды в Codespaces

1. Создайте новый Codespace в GitHub на основе официального образа с Python 3.10+.

2. Откройте терминал Codespace и установите необходимые зависимости:
   ```
   pip install numpy scipy librosa scikit-learn matplotlib torch torchaudio
   ```
3. Настройте структуру проекта:
   ```
   project/
   ├── data/        # аудиофайлы
   ├── features/    # сохранённые спектрограммы/признаки
   ├── models/      # модели и веса
   └── notebooks/   # Jupyter-ноутбуки для экспериментов
   ```

2. Получение и подготовка данных

Выбор датасета. Для быстрого MVP удобно использовать небольшой открытый набор, например **WaveFake**. Авторы собрали десять наборов примеров, полученных от шести разных вокодеров, и отметили, что такой набор позволяет сравнивать генераторы один-к-одному【782821870482790†L61-L75】. Альтернативой является часть **Logical Access** (LA) из соревнования ASVspoof 2019/2021.
Загрузите WaveFake с Zenodo (ссылка дана в статье) или возьмите LA-датасет. Сохраните аудио в `data/` и создайте метки (`0` — реальное, `1` — фейк).

3. Предобработка: извлечение признаков

Для большинства моделей аудиодипфейков используются спектральные признаки. WaveFake подробно описывает, как строить спектрограмму:
- **Фреймирование**: разделите сигнал на кадры (например, 20 мс) с частичным перекрытием (10 мс) и примените оконную функцию (Хэмминг, Ханн и т.д.) для снижения спектрального вытекания【782821870482790†L112-L123】.
- **Преобразование Фурье**: для каждого кадра вычислите дискретное преобразование Фурье и возьмите квадрат модуля, чтобы получить спектрограмму【782821870482790†L125-L132】.

- **Mel-фильтрбанк**: для **Mel-спектрограммы** примените треугольные фильтры по шкале Мел, соответствующие нелинейному восприятию частот человеком 【782821870482790†L133-L149】.

WaveFake также использует **MFCC** (Mel-частотные кепстральные коэффициенты), получаемые из логарифма Mel-спектрограммы и дискретного косинусного преобразования 【782821870482790†L179-L197】. Вместо них можно использовать **LFCC** (линейно-частотные коэффициенты) — они извлекаются так же, но с линейным фильтрбанком, что сохраняет больше высоких частот 【782821870482790†L200-L205】.

Чтобы извлечь признаки:

```
import librosa
import numpy as np

def extract_mfcc(path, sr=16000, n_mfcc=20, hop_length=512):
    y, _ = librosa.load(path, sr=sr)
    # MFCC (можно заменить librosa.feature.lfcc для LFCC)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc,
hop_length=hop_length)
    # Первые и вторые производные (дельта-признаки) улучшают представление
【782821870482790†L208-L213】.
    delta = librosa.feature.delta(mfcc)
    delta2 = librosa.feature.delta(mfcc, order=2)
    return np.vstack([mfcc, delta, delta2]).T
```

В альтернативной работе показано, что использование **лог-спектрограмм** или **CQT-спектрограмм** (постоянно-Q-преобразование) увеличивает качество в среднем на 37 % по EER по сравнению с Mel-спектрограммами 【254952313190576†L20-L28】. Для MVP можно попробовать оба типа:

```
# Лог-спектрограмма
import librosa

def extract_logspec(path, sr=16000, n_fft=1024, hop_length=512):
    y, _ = librosa.load(path, sr=sr)
    stft = librosa.stft(y, n_fft=n_fft, hop_length=hop_length)
    log_spec = np.log1p(np.abs(stft))
    return log_spec.T

# CQT-спектрограмма

def extract_cqt(path, sr=16000):
    y, _ = librosa.load(path, sr=sr)
    cqt = np.abs(librosa.cqt(y, sr=sr, n_bins=84))
    return librosa.amplitude_to_db(cqt).T
```

Сохраните извлечённые признаки (например, в `.npy`) в папке `features/`, чтобы не перерасчитывать их.

4. Базовая модель: гауссовые смеси (GMM)

Организаторы ASVspoof предоставляют две эталонные модели — **Gaussian Mixture Model (GMM)** и **RawNet2**【782821870482790†L668-L676】. GMM — лёгкий вариант для MVP:

1. Разделите набор на обучение (80 %) и тест (20 %).

2. Для каждого типа аудио (реальное/фейковое) обучите отдельную модель `GaussianMixture` из scikit-learn, используя MFCC или LFCC + дельты.

3. Для классификации вычислите **лог-отношение правдоподобия**: [(x) = p(x_n) - p(x_s),] где (_n) — параметры GMM для реального аудио, (_s) — для фейкового【782821870482790†L706-L718】. Положительные значения относим к классу «реальное», отрицательные — к «фейк».

4. Используйте **Equal Error Rate (EER)** для оценки. EER — это точка на кривой ROC, где частоты ложного принятия и ложного отклонения равны【782821870482790†L693-L704】.

Пример кода:

```
from sklearn.mixture import GaussianMixture
from sklearn.metrics import roc_curve

# X_real, X_fake – матрицы признаков для реального и фейкового аудио
+gmm_real = GaussianMixture(n_components=16,
covariance_type='diag').fit(X_real)
+gmm_fake = GaussianMixture(n_components=16,
covariance_type='diag').fit(X_fake)
+
+def log_likelihood_ratio(X):
+    return gmm_real.score_samples(X) - gmm_fake.score_samples(X)
+
+def compute_eer(scores, labels):
+    fpr, tpr, _ = roc_curve(labels, scores)
+    fnr = 1 - tpr
+    idx = np.argmin(np.abs(fpr - fnr))
+    return max(fpr[idx], fnr[idx])
+
+scores = np.hstack([log_likelihood_ratio(X_real_test),
+                    log_likelihood_ratio(X_fake_test)])
+labels = np.hstack([np.ones(len(X_real_test)), np.zeros(len(X_fake_test))])
+eer = compute_eer(scores, labels)
+print(f"EER: {eer:.3f}")
```

Исследование WaveFake показывает, что при использовании LFCC-признаков GMM-классификатор показывает лучшие результаты, чем с MFCC【782821870482790†L706-L723】.

5. Базовая нейронная модель: RawNet2

Для более сильного бейзлайна можно использовать **RawNet2**. Это гибрид CNN-GRU, который извлекает эмбеддинг из сырого аудио и затем обучает полносвязный слой для задачи детекции【782821870482790†L724-L729】. Вы можете:

- Взять готовую реализацию RawNet2 (например, из репозитория ASVspoof) и запустить обучение в Codespace.
- Сравнить EER модели с GMM на тех же данных.

6. Улучшенные признаки и архитектуры

Исследование **Does Audio Deepfake Detection Generalize?** отмечает, что выбор признаков сильно влияет на успех. Использование CQT-спектрограммы или лог-спектрограммы вместо Mel-спектрограммы увеличивает среднее качество на 37 % по EER【254952313190576†L20-L28】. Попробуйте заменить MFCC/LFCC на эти признаки (см. примеры функций выше).
Также исследование предлагает испытать разные архитектуры: простую LSTM, LCNN, ResNet18, Transformer, CRNNSpoof, RawNet2 и др.【254952313190576†L64-L131】. Для MVP достаточно сравнить GMM и одну нейронную модель.

7. Проверка обобщаемости

Авторы работы *Does Audio Deepfake Detection Generalize?* показали, что модели, обученные на ASVspoof, плохо распознают дипфейки «в дикой среде». Они собрали 37,9 часа записей известных людей (из них 17,2 часа — дипфейки) и обнаружили, что качество на реальных данных деградирует до тысячи процентов【254952313190576†L20-L31】. После обучения на WaveFake/ASVspoof, загрузите пару реальных дипфейков (например, из открытых источников) и оцените модель, чтобы понять её обобщаемость.

8. Дополнительные возможности

Для более продвинутого MVP вы можете попробовать **wav2vec 2.0**. Эта модель учится на сыром аудио в самосупервизированном режиме, затем дообучается на небольшой размеченной выборке【455095852628783†L9-L24】. Используйте предобученную модель из `torchaudio` и обучите классификатор на извлечённых эмбеддингах. Пример:

```
import torchaudio, torch

bundle = torchaudio.pipelines.WAV2VEC2_BASE
model = bundle.get_model()

def extract_wav2vec_features(path):
    waveform, sr = torchaudio.load(path)
    if sr != bundle.sample_rate:
```

```
        waveform = torchaudio.functional.resample(waveform, sr,
bundle.sample_rate)
    with torch.no_grad():
        features = model.extract_features(waveform)[0].squeeze(0).numpy()
    return features
```

Итог

- Начните с базового набора WaveFake и реализуйте простую GMM-модель для быстрой проверки.

- Используйте LFCC-признаки и дельта-коэффициенты; это даёт надёжную отправную точку【782821870482790†L706-L723】.

- Постепенно экспериментируйте с лог- и CQT-спектрограммами, а также с нейронной моделью RawNet2; исследования показывают, что выбор признаков может снижать EER до 0,062 при обучении на FB-MelGAN【782821870482790†L748-L752】.

- Оцените модель на реальных дипфейках; научные работы подчёркивают, что результаты на лабораторных и реальных данных сильно различаются【254952313190576†L23-L31】.

Следуя этим шагам, вы построите минимальный, но обоснованный MVP для задачи аудио-дипфейк-детекции и сможете дальше улучшать модель.