Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА ____СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

по дисциплине _____Технологии машинного обучения_____

по теме_____«Удаление шумов из аудио»_____

_____

_____

_____

Студентка ИУ5-63Б _____ В.В. Шаповалова
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ А.И. Канев
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2023 г.

Аннотация

По ходу курса предмета «Технологии машинного обучения» было поставлено задание по созданию нейросети, способной удалить шумы из аудио. Исследование можно разделить на три части работы: разведочный анализ, предварительная обработка данных в питоне, создание алгоритма. В ходе работы будут использоваться такие библиотеки как: «TensorFlow» и «Keras». Анализ будет проводиться с помощью программы «Jupyter Python».

Будет производиться обучение модели и построены соответствующие графики и таблицы.

По итогу работы будет выполнен разведочный анализ а также создан свой алгоритм машинного обучения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – обучение модели для удаления шумов из аудио.

Задачи:

1) Выполнить разведочный анализ, изучить актуальные алгоритмы для похожих задач;
2) Выполнить первичную обработку данных, преобразовать аудиофайлы в спектрограммы, а их в свою очередь в массивы с нормализованными значениями;
3) Обучить собственную модель для генерации чистых аудио на основе аудио с шумом в Jupyter Notebook, получить информацию о точности работы данной модели и проанализировать результаты;

1. Постановка задачи

Чтобы поставить задачу, нужно ознакомиться с метриками, используемыми для сепарации речи. Нужные метрики зависят от конкретной задачи, но их можно разделить на 2 класса: уровня сигнала и уровня восприятия. На уровне сигнала метрики направлены на количественную оценку степени усиления сигнала или уменьшения помех. [3]

Модели улучшения речи не работают непосредственно с необработанным сигналом (т.е. сигналом во временной области), скорее они включают дискретное преобразование Фурье (DFT) в свой конвейер обработки сигналов, главным образом, в качестве первого шага для преобразования сигнала во временной области в частотную область. Эти модели признают, что речевые сигналы сильно не стационарны и их характеристики изменяются как по времени, так и по частоте. Следовательно, извлечение их частотно-временных характеристик с помощью DFT позволит лучше передать представление сигнала речевого сигнала [4].

Для отделения речи от посторонних шумов, без потери смысловой нагрузки и качества, в новых алгоритмах чаще всего используют метод градиента.

Наиболее актуальный алгоритм, представленный на рисунках 1.1 и 1.2, – преобразовать тренировочные аудиофайлы с шумами и без в waveform или спектрограммы, наложить графики друг на друга, использовать кодировщикии (encoder), декодировщики (decoder), а также сети для сепарации речи (SS Net) и усиления речи (SE Net), чтобы получить оптимальную маску шумного и чистого аудиофайлов [1].

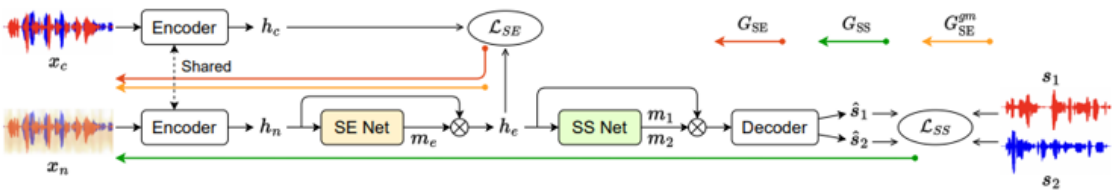

Рисунок 1.1 – Архитектура модели обучения

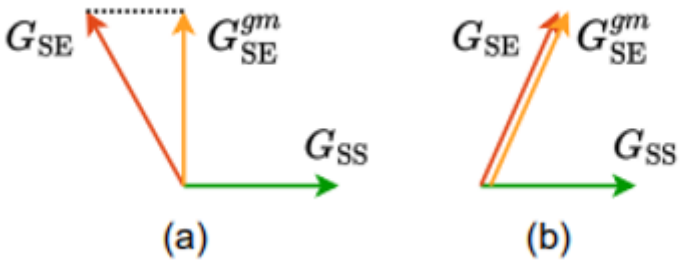

Рисунок 1.2 – Модуляция градиентов

Мне предстоит решить задачу, где есть один аудио канал, но сложнее – когда начальные данные - микс. Не так давно была создана база данных (WHAM!), в которой есть 2 канала: по одному для говорящего, а также, наложен микс звуков, которые имитируют реальную окружающую среду. Для решения задачи, где нужно отделить голоса людей и убрать шум, используют больше разновидностей исходных файлов. Естественно, в таком случае для сравнения используют более сложные функции и методы визуализации, но такой вариант наиболее приближен к реальной жизни. [2]

Обычно в таких случаях используют спектрограммы, поскольку они отражают изменение в частоте сигнала в течении времени. [5]

2. Разведочный анализ

Поскольку в моей задаче всего один источник шума и одна дорожка голоса, я выбрала алгоритм, представленный на рисунке 2 для решения задачи.[4]

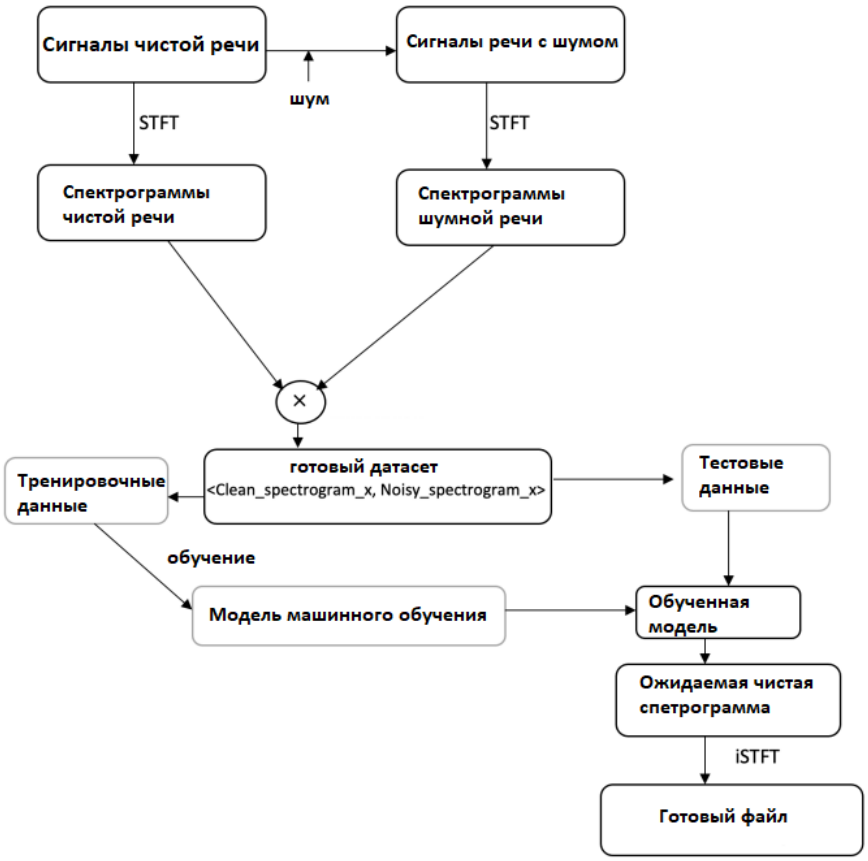

Рисунок 2 – Алгоритм решения задачи

Учитывая сигналы смешанной (зашумленной) речи, преобразуем необработанные сигналов зашумленной речи в спектрограмму. Преобразуем соответствующий сигнал чистой речи во временной области в то же представление, что и для зашумленной речи. Обучаем модель глубокого обучения, чтобы узнать, как оценивать чистые функции с учетом шумных речевых функций в качестве входных данных путем минимизации целевой функции. Учитывая новые шумные речевые характеристики, обученная модель должна оценить чистые речевые характеристики. Используя оцененные признаки чистой речи нужно восстановить ее необработанную форму сигнала, выполнив процесс, обратный процессу генерации признаков (например,

используя обратное кратковременное преобразование Фурье, если признаки находятся во частотно-временной области). SFT на схеме как раз показывает, где к файлу применяется кратковременное преобразование Фурье, а iSFT – обратное преобразование.

Обучение состоит из трех этапов: кодирование, оценка маски или разделение и декодирование. Кодировщик преобразует входные смешанные сигналы во временной области в промежуточное представление, используя слои свёртки.

Оценщик масок вычисляет одну маску для говорящего. Промежуточное представление динамика получается путем умножения выходных данных кодера на его соответствующую маску. Оценщик маски состоит из блоков свертки уровня и полносвязного слоя.

Декодер преобразует промежуточные представления в разделенные во временной области речевые сигналы, используя транспонированные свёртки слои. [7].

3. Удаление шумов из аудио

Первичная обработка данных заключалась в преобразование аудио датасета в датасет спектрограмм. Примеры спектрограмм одного и того же аудиофайла с шумом и без представлены на рисунках 3.1 и 3.2.

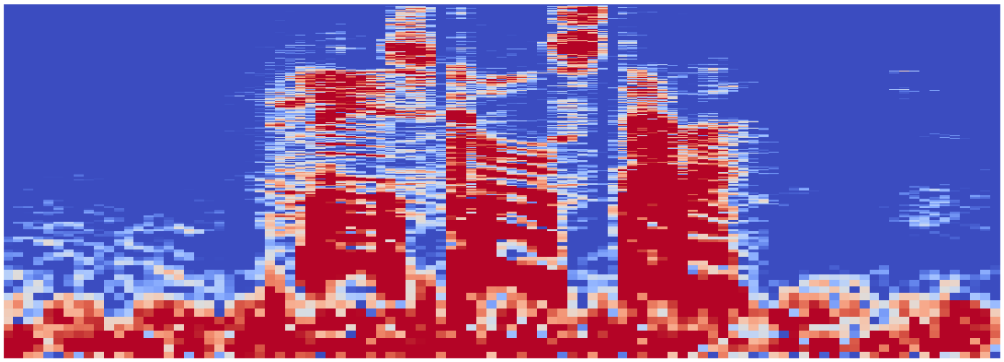

Рисунок 3.1 – Спектрограмма аудио без шума

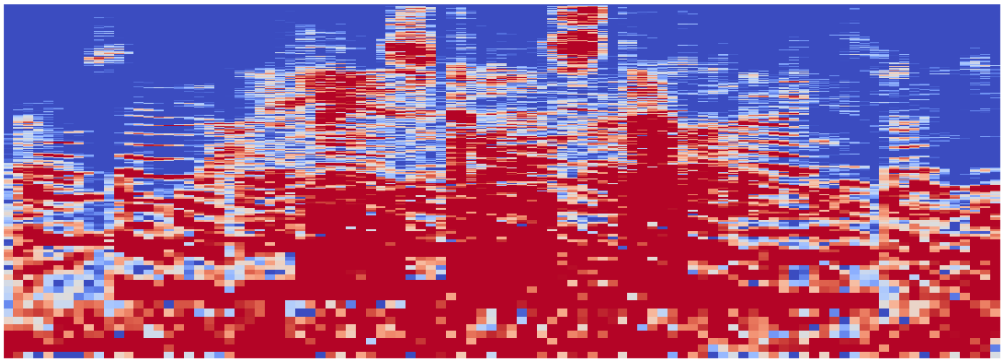

Рисунок 3.2 – Спектрограмма аудио с шумом

На рисунке 4 опишем функцию. которая будет открывать изображения и преобразовывать их в одинаковый размер для обучения модели, а также представлять изображение в виде массива с нормализованными значениями пикселей.

```
# Путь к датасету
dataset_path = 'spectograms_dataset'

newsize = (128, 256)
# Функция, которая заргужает изображение, меняет его размер и преобразует в массив питру
def load_image(file_path):
    img = Image.open(file_path)
    #img = img(lambda x, y: (x / 255, y))
    img =img.resize(newsize)
    img_arr = np.array(img)
    img_arr = (img_arr - 127.5) / 127.5  # Нормализация значений пикселей
    return img_arr
```

Рисунок 4 – Преобразование спектрограммы

На рисунке 5 создадим поток, который будем подавать на вход для обучения модели – спектрограммы шумных аудио файлов

```
# Датасет "До" фото
X_train = []
for file_name in os.listdir(os.path.join(dataset_path, 'noisy')):
    file_path = os.path.join(dataset_path, 'noisy', file_name)
    img_arr = load_image(file_path)
    if len(X_train)<100:
        X_train.append(img_arr)
X_train = np.array(X_train)
```

Рисунок 5 – Создание потока данных с шумом

На рисунке 6исоздадим поток, который будем подавать на выход для обучения модели –спектрограммы чистых аудио файлов

```
# Датасет "После" фото
Y_train = []
for file_name in os.listdir(os.path.join(dataset_path, 'clean')):
     file_path = os.path.join(dataset_path, 'clean', file_name)
     img_arr = load_image(file_path)
     if len(Y_train)<100:
        Y_train.append(img_arr)
Y_train = np.array(Y_train)
```

Рисунок 6 – Создание потока данных без шума

Создадим сверточную нейронную сеть – рисунок 7. В данной модели слои кодировщика и декодировщика применяют умножение матрицы свертки с входным тензором только на одной оси, вычисляя свертку вдоль оси времени. А

также есть связывающий слой, где каждый нейрон связан со всеми нейронами предыдущего слоя.

```
input_shape = (500, 1400, 4)
input_img  = Input(shape=input_shape)
# Reshape the input into a 1-Dimensional sequence
#x = Reshape((256*128, 4))(input_img)

# Encoder layers
x = Conv2D(32, 4, activation='relu', padding='same')(input_img)
x = Conv2D(64, 4, activation='relu', padding='same')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)
x = Conv2D(128, 4, activation='relu', padding='same')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)
x = Conv2D(256, 4, activation='relu', padding='same')(x)


#x = Flatten()(x)
x = Dense(4, activation='relu')(x)

# Decoder layers
x = Conv2D(128, 4, activation='relu', padding='same')(x)
x = UpSampling2D(size=(2, 2))(x)
x = Conv2D(64, 4, activation='relu', padding='same')(x)
x = UpSampling2D(size=(2, 2))(x)
x = Conv2D(4, 4, activation='sigmoid', padding='same')(x)

# Reshape the output back into an image format
decoded = Reshape((500, 1400, 4))(x)
denoise_autoencoder = Model(input_img, decoded)
```

Рисунок 7 – Создание нейросети

На рисунке 8 компилируем нейронную сеть с функцией потерь «средняя квадратичная ошибка», которая в данном контексте используется для измерения расхождения между исходным шумным изображением и выходным "очищенным" изображением, которое было сгенерировано автокодировщиком.

```
denoise_autoencoder.compile(optimizer="adam", loss="mse")
```

Рисунок 8 – Компиляция нейросети

На рисунке 9 выводим описание архитектуры нейронной сети, с указанием количества параметров и связей между слоями.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 500, 1400, 4)]    0

 conv2d (Conv2D)             (None, 500, 1400, 32)     2080

 conv2d_1 (Conv2D)           (None, 500, 1400, 64)     32832

 max_pooling2d (MaxPooling2D  (None, 250, 700, 64)     0
 )

 conv2d_2 (Conv2D)           (None, 250, 700, 128)     131200

 max_pooling2d_1 (MaxPooling  (None, 125, 350, 128)    0
 2D)

 conv2d_3 (Conv2D)           (None, 125, 350, 256)     524544

 dense (Dense)               (None, 125, 350, 4)       1028

 conv2d_4 (Conv2D)           (None, 125, 350, 128)     8320

 up_sampling2d (UpSampling2D  (None, 250, 700, 128)    0
 )

 conv2d_5 (Conv2D)           (None, 250, 700, 64)      131136

 up_sampling2d_1 (UpSampling  (None, 500, 1400, 64)    0
 2D)

 conv2d_6 (Conv2D)           (None, 500, 1400, 4)      4100

 reshape (Reshape)           (None, 500, 1400, 4)      0

=================================================================
Total params: 835,240
Trainable params: 835,240
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 8 – Архитектура нейросети

На рисунке 9 определяем куда сохранять логи

```
from keras.callbacks import TensorBoard
logdir2 = 'logs'
tensorboard_callback = TensorBoard(log_dir=logdir2)
```

Рисунок 9 – Сохранение логов

На рисунке 10 обучаем нейросеть, разбиваем обучение на 4 эпохи, разбиваем данные на тренировочные и данные для валидации. На вход и выход подаём ранее сформированные данные.

```
hist = denoise_autoencoder.fit(x=x_train, y=y_train,
                               shuffle=False, epochs=4,
                               callbacks= [tensorboard_callback],
                               validation_data=(x_test, y_test))
```

Рисунок 10 – Обучение нейросети

На рисунке 11 получаем значение метрик для каждой эпохи обучения.

```
In [34]: hist.history

Out[34]: {'loss': [0.1758790761232376,
           0.17242774367332458,
           0.16780897974967957,
           0.1619076430797577],
          'val_loss': [0.17375293374061584,
           0.16939625144004822,
           0.1640285849571228,
           0.1567690521478653]}
```

Рисунок 11 – Значение метрик

На рисунках 12.1 и 12.2. строим график функции потерь при обучении и валидации.

```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
fig =plt.figure()
plt.plot(hist.history['loss'], color ='green', label ='loss')
plt.plot(hist.history['val_loss'], color ='purple', label ='val_loss')
plt.suptitle('Loss')
plt.legend(loc='upper left')
plt.show()
```

Рисунок 12.1 – Данные для графика

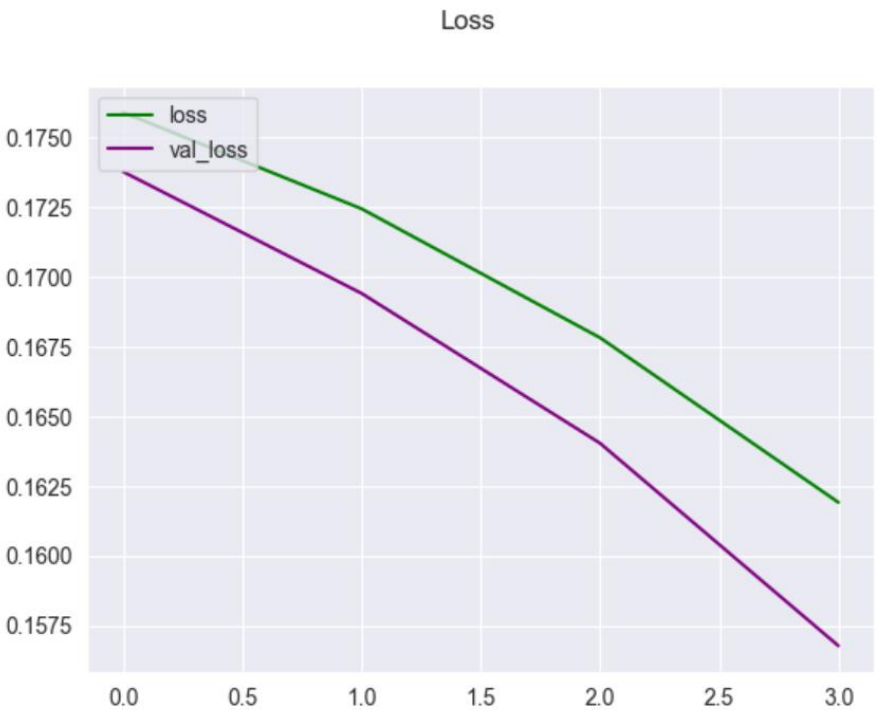

Рисунок 12.1 – График функции потерь

На рисунке 13 тестируем нейросеть на новой спектрограмме и сохраняем результат в формате фото.

```
import numpy as np
from PIL import Image

# Load the image
img = Image.open("n.png")
newsize = (500, 1400)
# Convert the image to RGB mode

img = img.resize(newsize)
img = img.transpose(Image.TRANSPOSE)  # Transpose the image dimensions
img_array = np.array(img)
img_array = np.expand_dims(img_array, axis=0)

# Normalize the pixel values
img_array = img_array / 255.0

# Make prediction using the pre-trained model
denoised_img_array = denoise_autoencoder.predict(img_array)

# Scale the pixel values back to the range of 0-255
denoised_img_array = (denoised_img_array * 255.0).astype(np.uint8)

# Convert the numpy array back to an image and save it
denoised_img = Image.fromarray(denoised_img_array[0])
denoised_img.save("denoised_audio.png")
```

Рисунок 13 – Тестирование нейросети

На рисунках 14.1, 14.2, 14.3 представлены результаты работы нейросети

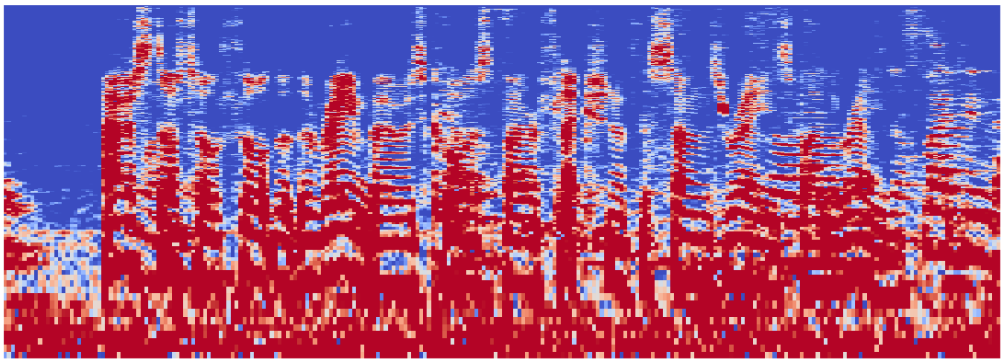

Рисунок 14.1 – Шумная спектрограмма на входе

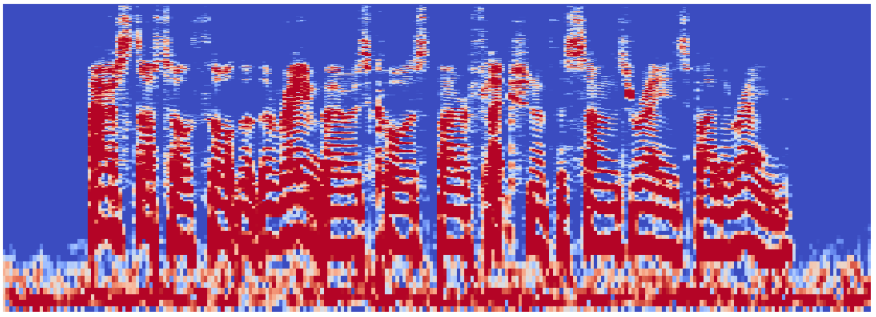

Рисунок 14.2 – Результат нейросети

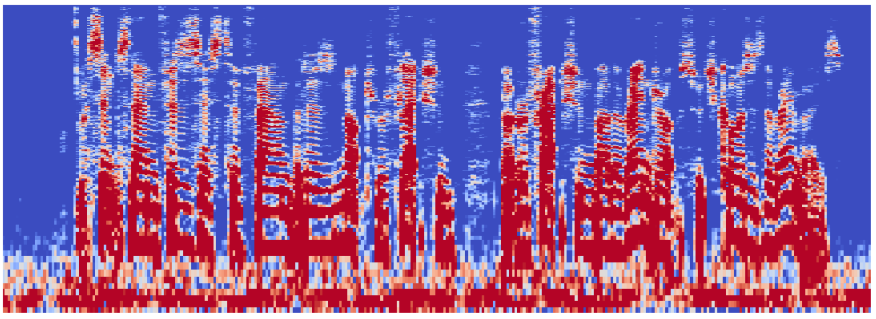

Рисунок 14.2 – Исходная спектрограмма без шума

ЗАКЛЮЧЕНИЕ

В ходе научно-исследовательской работы был проведен анализ кейса «Удаление шумов из аудиофайлов», была обучена модель генерации чистого аудио на основе аудио с шусом:

1) Были изучены актуальные решения данной задачи и выбран оптимальный алгоритм для данного кейса

2) Выполнена первичная обработка данных, преобразован датасет аудио формата в данные, подходящие для обучения сверточной нейронной сети.

3) Была обучена собственная модель для удаления шума из аудиофайлов в Jupyter Notebook, точность алгоритма не минимальная, но может быть улучшена за счёт большего количества тестовых данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Yuchen Hu, Chen Chen, Heqing Zou, Xionghu Zhong, Eng Siong Chng, «Unifying Speech Enhancement and Separation with Gradient Modulation for End-to-End Noise-Robust Speech Separation», Accepted by ICASSP 2023, https://arxiv.org/abs/2302.11131

2. Gordon Wichern, Joe Antognini, Michael Flynn, Licheng Richard Zhu, Emmett McQuinn, Dwight Crow, Ethan Manilow, Jonathan Le Roux «WHAM!: Extending Speech Separation to Noisy Environments», Accepted for publication at Interspeech 2019, https://arxiv.org/abs/1907.01160

3. DeLiang Wang, Jitong Chen, «Supervised Speech Separation Based on Deep Learning: An Overview», 15 Jun 2018, https://arxiv.org/abs/1708.07524

4. Peter Ochieng, «Deep neural network techniques for monaural speech enhancement: state of the art analysis», 1 Dec 2022 https://arxiv.org/abs/2212.00369

5. Yi Luo, Nima Mesgarani, «Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation», Accepted by IEEE/ACM Transactions on Audio, Speech and Language Processing 2019, https://arxiv.org/abs/1809.07454

6. Ruohan Gao1,2 Kristen Grauman1,3 1The University of Texas at Austin 2Stanford University 3Facebook AI Research, « VISUALVOICE: Audio-Visual Speech Separation with Cross-Modal Consistency», In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021

7. MathWorks, End-to-End Deep Speech Separation, https://www.mathworks.com/help/deeplearning/ug/end-to-end-deep-speech-separation.html