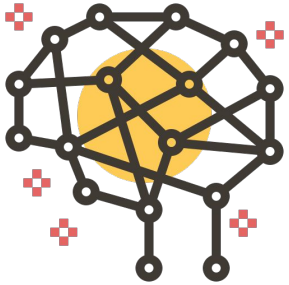
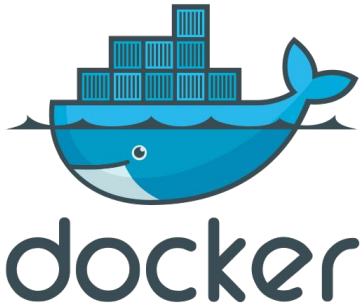


SER: Speech Emotion Recognition

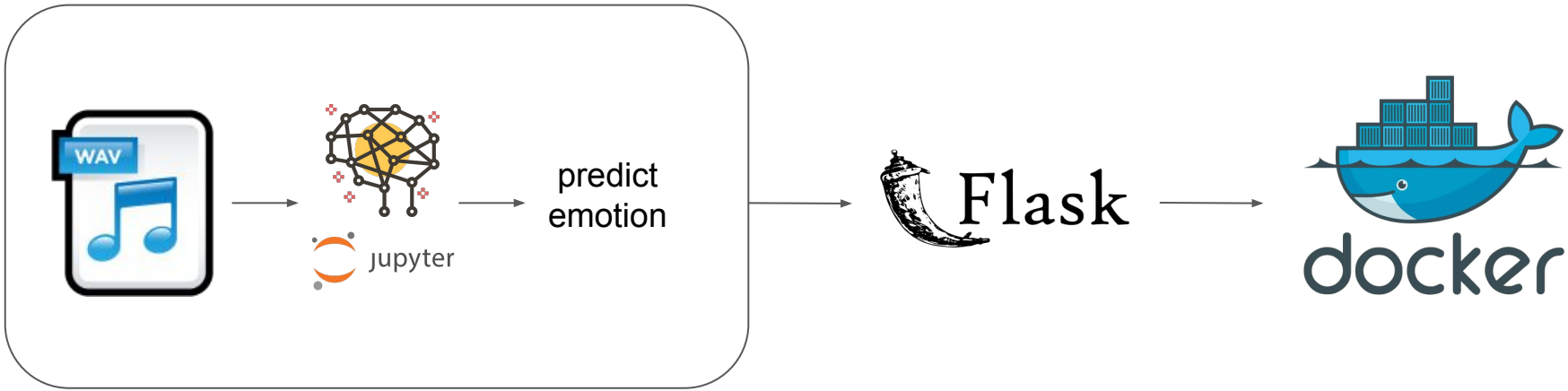
Marija Stojchevska



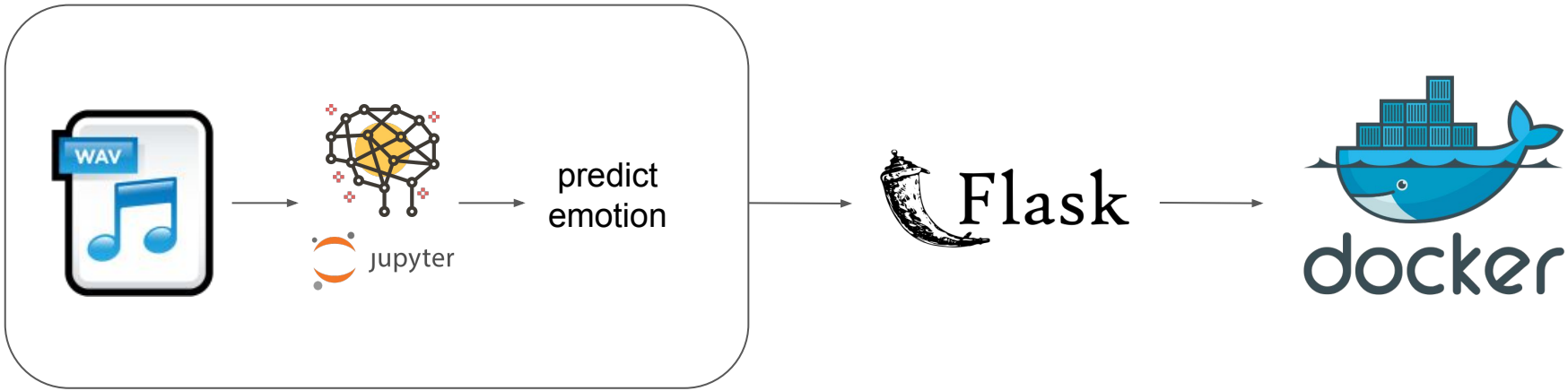
Agenda

1	Goal
2	Dataset
3	Model
4	Docker
5	Flask API
6	Future Work

Goal



Goal



- Docker Container exposes the Jupyter notebook and Flask API on different ports.

Agenda

1	Goal
2	Dataset
3	Model
4	Docker
5	Flask API
6	Future Work

Dataset - Emo DB

03a04Ad.wav

emotion class

Dataset - Emo DB

Audio files

03a04Ad.wav
09b09Nd.wav
.
.
.
.

11a04Fd.wav

Dataset - Emo DB

Audio files

03a04Ad.wav
09b09Nd.wav
.
.
.
.
11a04Fd.wav

Dictionary of the emotions

```
char_to_emotion = {
    'W': 'anger',
    'L': 'boredom',
    'E': 'disgust',
    'A': 'fear',
    'F': 'happiness',
    'T': 'sadness',
    'N': 'neutral',
}
```

Dataset - Emo DB

Audio files

03a04Ad.wav
09b09Nd.wav
.
.
.
.
11a04Fd.wav

Dictionary of the emotions

```
char_to_emotion = {
    'W': 'anger',
    'L': 'boredom',
    'E': 'disgust',
    'A': 'fear',
    'F': 'happiness',
    'T': 'sadness',
    'N': 'neutral',
}
```

Data frame: audio to emotion

	emotions
03a01Fa.wav	happiness
03a01Nc.wav	neutral
03a01Wa.wav	anger
03a02Fc.wav	happiness
03a02Nc.wav	neutral
...	...
16b10Lb.wav	boredom
16b10Tb.wav	sadness
16b10Td.wav	sadness
16b10Wa.wav	anger
16b10Wb.wav	anger

Dataset - class distribution

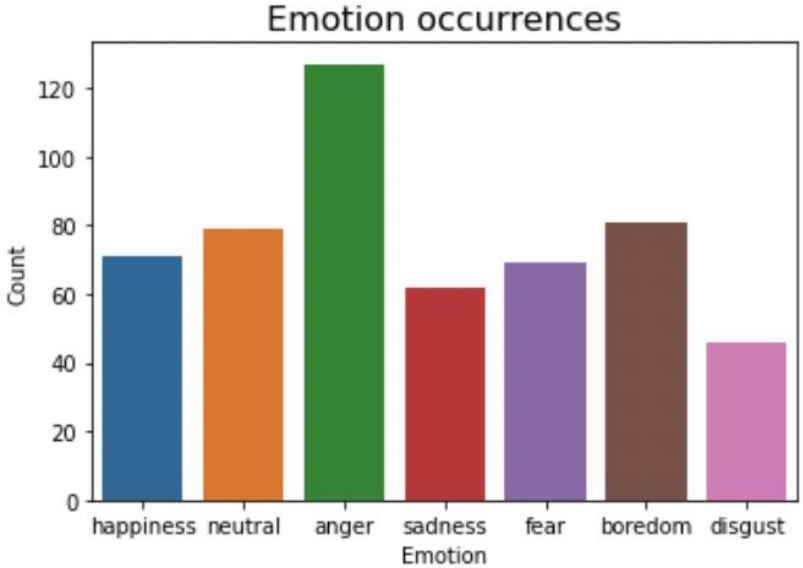
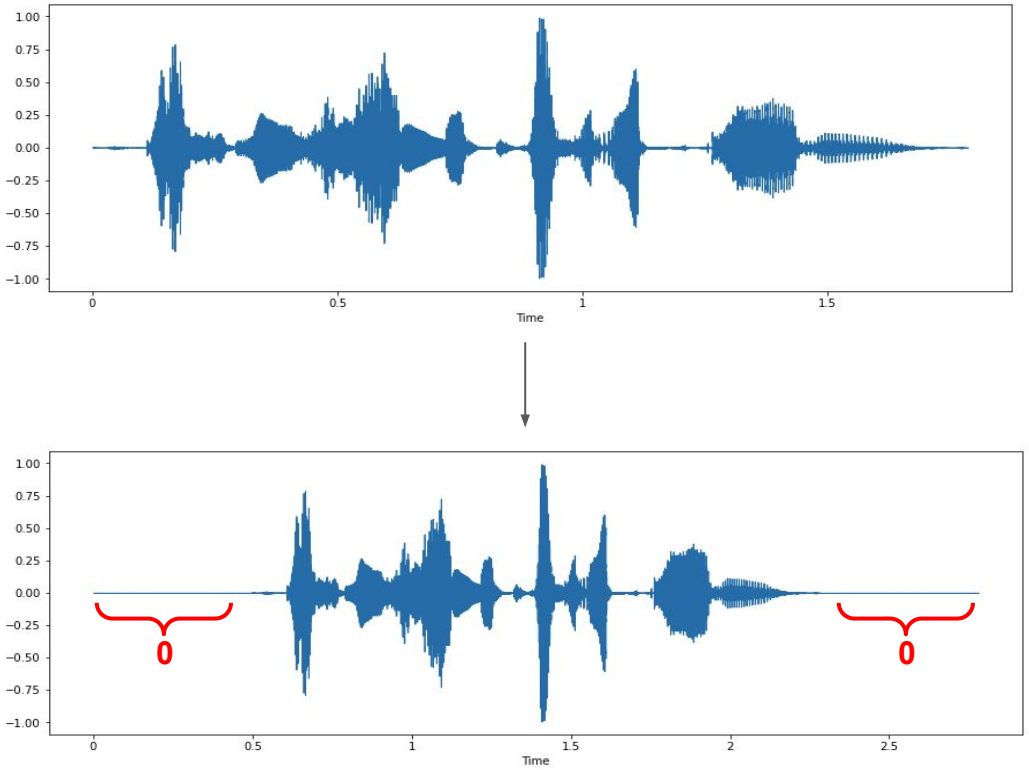


Figure 1. Class distribution histogram. The seven categories are plotted on the x-axis, while the y-axis represents the number of occurrences for each category.

Dataset - data preprocessing

Zero padding

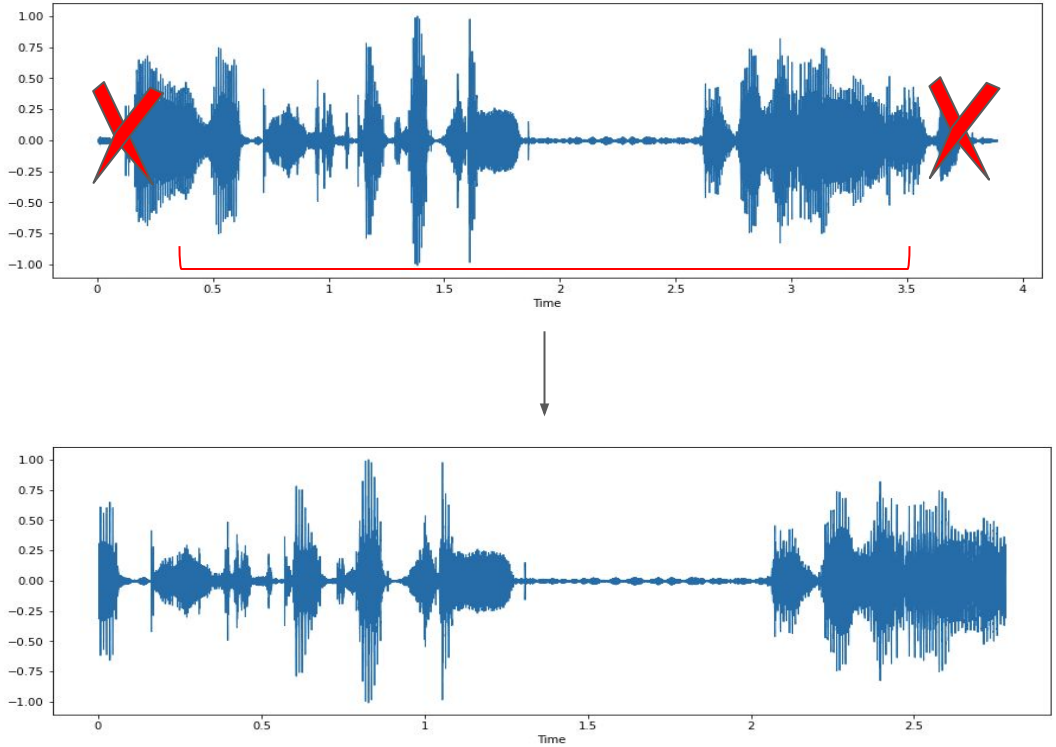
Satisfy the average number of samples over all audio recordings in the dataset



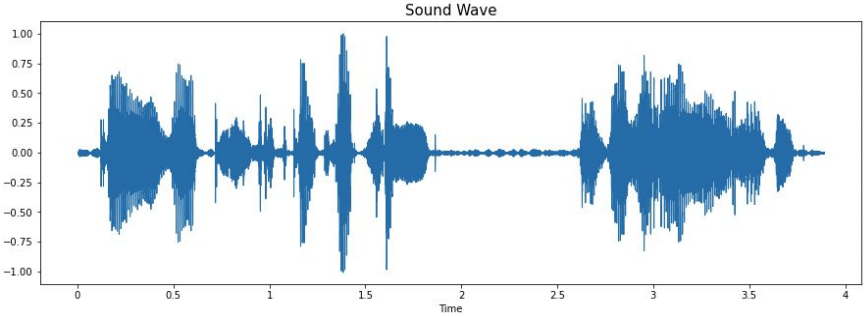
Dataset - data preprocessing

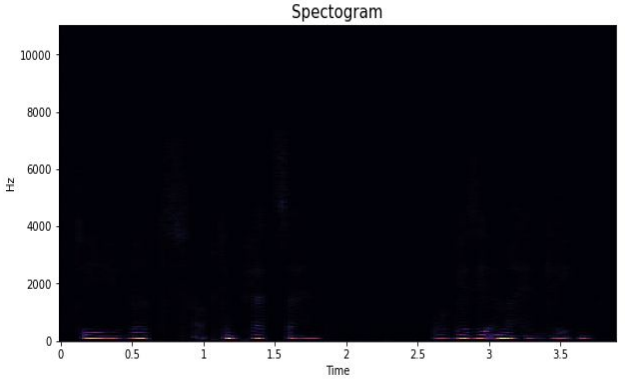
Length cutting

Satisfy the average number of samples over all audio recordings in the dataset

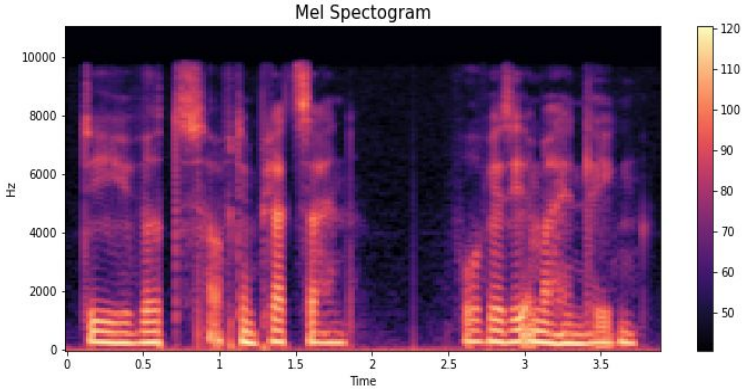


Dataset - feature extraction

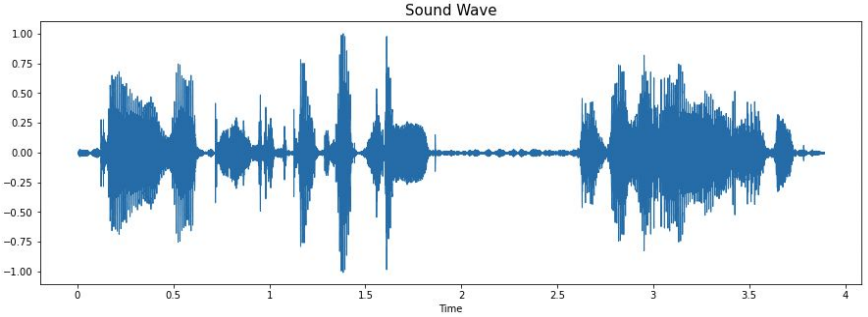


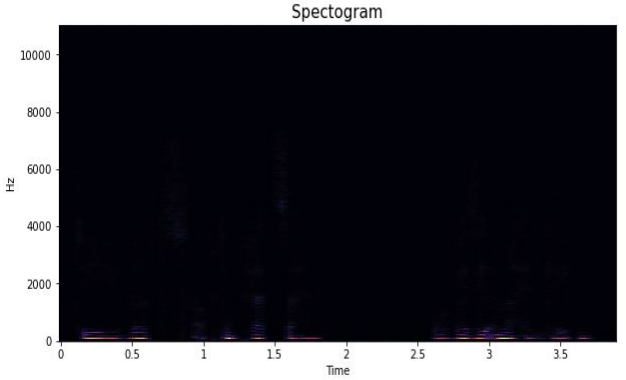


Melspec features

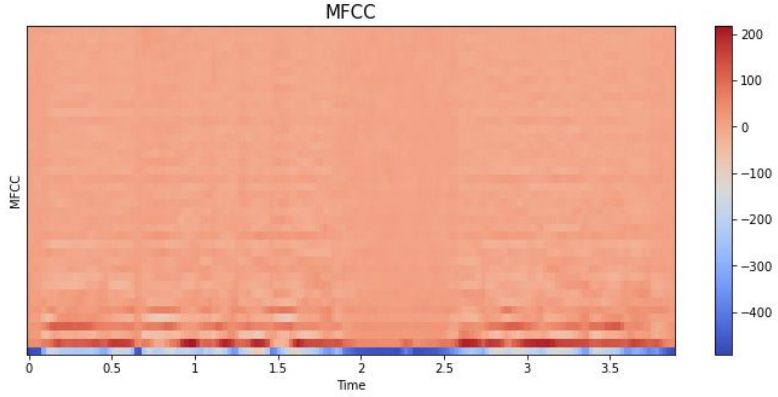


Dataset - feature extraction





Mfcc features



Dataset - labels preprocessing

- Handling categorical variables with one-hot encoding

	emotions
03a01Fa.wav	happiness
03a01Nc.wav	neutral
03a01Wa.wav	anger
03a02Fc.wav	happiness
03a02Nc.wav	neutral

```
OneHotEncoder()
[[0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0.]
 [1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0.]
```

Agenda

1	Goal
2	Dataset
3	Model
4	Docker
5	Flask API
6	Future Work

Model - Convolutional Neural Network

```
model.add(Conv1D(filters=64, kernel_size=5, padding="same",activation="relu", input_shape=(X_train.shape[1],1)))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv1D(filters=125, kernel_size=10, padding="same",activation="relu"))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv1D(filters=64, kernel_size=5, padding="same",activation="relu"))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(7, activation="relu"))
model.add(BatchNormalization())
model.add(Dropout(0.3))
model.add(Dense(7, activation="softmax"))
```

Figure 4. Model architecture.

Agenda

1	Goal
2	Dataset
3	Model
4	Docker
5	Flask API
6	Future Work

Docker - dockerfile

```
FROM python:3.9

RUN mkdir /app
COPY . /app
WORKDIR /app

RUN apt-get update \
&& apt-get upgrade -y \
&& apt-get install -y \
&& apt-get -y install apt-utils gcc libpq-dev libsndfile-dev

RUN pip install jupyterlab
RUN pip install -r libraries.txt

EXPOSE 105
```

Figure 3. Content of the Dockerfile.

Docker - .yaml file

```yaml
version: '3.8'
services:

  report:
    build: .
    ports:
      - "8888:8888"
    volumes:
      - ./SEReport:/app
    entrypoint:
      jupyter notebook --ip='0.0.0.0' --port=8888 --no-browser --allow-root --NotebookApp.token='' --NotebookApp.password=''

  api:
    build: .
    ports:
      - "8000:8000"
    expose:
      - 105
    volumes:
      - ./FlaskAPI:/app
    entrypoint:
      jupyter lab --ip='0.0.0.0' --port=8000 --no-browser --allow-root --NotebookApp.token='' --NotebookApp.password=''
```

Figure 4. Content of the .yaml file.

Agenda

1	Goal
2	Dataset
3	Model
4	Docker
5	Flask API
6	Future Work

Flask API

```
app = Flask(__name__)

# Endpoint for model training.
@app.route('/<int:train>/')
def model_training(train):
    if train==1:
        dataset_dir = download_and_extract(source, src_dir, dir_name)
        audio_files, audio_directory, emotion_df, emotions = dataset_reading(src_dir, dataset_dir)
        class_histogram(emotion_df, emotions)
        X_train, X_test, Y_train, Y_test, X_val, Y_val, encoder = dataset_split(audio_files,
                                                audio_directory, features_type, emotion_df, test_split, val_split)
        dump(encoder, 'encoder.joblib')
        model, history = CNN(X_train, Y_train, X_val, Y_val, batch_size, learning_rate, num_epochs, patience)
        evaluate(model, history, encoder, X_train, X_test, Y_train, Y_test, X_val, Y_val)
        save_model(model, src_dir)
        return "The newly trained model is saved in the my_model directory."
    else:
        return "To train the model type enter 1 at the end of the url. \n
        If the model is already trained, in the url variable section, \n
        you can enter the name of the audio file for which you want to predict the emotion."


# Endpoint for querying the last trained model with an audio file of our choice.
@app.route('/<string:name>/')
def emotion_prediction(name):

    model = tf.keras.models.load_model('.'+src_dir+'my_model')
    encoder = load('encoder.joblib')

    sound = '.'+src_dir+dir_name.split('.')[0]+'/wav/'+name
    samples, sr = librosa.load(sound)

    if features_type == 'mfcc':
        values = extract_mfcc(samples, sr)
    else:
        values = extract_melspec(samples, sr)

    x = pd.DataFrame([values]).iloc[:,:].values
    x = np.expand_dims(x, axis=2)

    # Predict emotion for one audio sample
    prediction = model.predict(x)
    predClass = encoder.inverse_transform(prediction)
    return "Predicted emotion for the audio file " + name + " is: " + predClass[0][0]


if __name__ == "__main__":
    app.run(host='0.0.0.0', port=port_number)
```

Figure 5. The relevant Flask API endpoints.

Agenda

1	Goal
2	Dataset
3	Model
4	Docker
5	Flask API
6	Future Work

Future Work

Dataset improvements:

- Try different preprocessing techniques
- Data augmentation
- Train the model on different input features

Model improvements:

- Reduce / increase the number of layers
- Play with different model hyperparameters
- RNN / LSTMs

Thank you
for your attention.