Assignment #4: Environmental Sound Classification

Team Members

- **Omar Hani Ibrahim Bishr 21010891**
- **Mohamed Mohamed Ibrahim Ali Hassan 21011211**
- **Mohamed Hassan Mohamed Qabary Hassan 21011115**

Feature Space Part

Approach:

1. Adding all train folds in 1 csv (which contains class and path columns).
2. Plotting waveform and spectrogram of a sample in the dataset.
3. Feature extraction of the dataset
    a. Features
        i. Energy
        ii. Mel Spectrogram
        iii. Mfcc
    b. Using data augmentation for training dataset
    c. Using padding with/without masking for variable length audio
4. Different combinations of features
    a. 1st option → 3 features per audio clip
        i. Energy
        ii. Mean of Mel Spectrogram

iii. Mean of Mfcc

b. 2nd option → 54 features per audio clip (what is used)
    i. Energy
    ii. Mel Spectrogram → 40 Mel bands
    iii. Mfcc → 13 coefficients
c. 3rd option → 13 features per audio clip
    i. Mfcc → 13 coefficients (without masking)
d. 4th option → 13 features per audio clip
    i. Mfcc → 13 coefficients (with masking)

Results:

1. All of the previous data caused overfit using simple GRU model

Building the Model

LSTM Model

Approach

To evaluate the impact of architectural and hyperparameter choices on time-series classification, we implemented four deep learning model variants using Keras. The key components across all models include:

1. Input Projection Layer

Each model begins with a TimeDistributed(Dense(...)) layer. This acts as a learnable projection that transforms the input features at each time step to a lower or higher-dimensional space (controlled by projection_dim). It serves as preprocessing for the recurrent or convolutional layers.

---

2. LSTM-Based Models

```
model = Sequential([
  TimeDistributed(Dense(projection_dim, activation='linear'), input_shape=input_shape),
  LSTM(lstm_units, return_sequences=True),
  LSTM(lstm_units, return_sequences=False),
  Dense(dense_units, activation='relu'),
  Dropout(dropout_rate),
  Dense(num_classes, activation='softmax')
])
```

- Two stacked LSTM layers are used.
- First LSTM returns sequences to feed into the second.
- Fully connected layers classify the final LSTM output.
- Dropout is applied before the final softmax layer to reduce overfitting.

Bonus

3. LSTM + CNN Hybrid

```
model = Sequential([
  TimeDistributed(Dense(projection_dim, activation='linear'), input_shape=input_shape),
  Conv1D(filters=64, kernel_size=3, activation='relu'),
  MaxPooling1D(pool_size=2),
  LSTM(lstm_units, return_sequences=True),
  LSTM(lstm_units, return_sequences=False),
  Dense(dense_units, activation='relu'),
  Dropout(dropout_rate),
  Dense(num_classes, activation='softmax')
])
```

- Applies a 1D convolution over time steps to extract local temporal patterns.
- The CNN output is downsampled via MaxPooling1D, then passed to stacked LSTMs.
- Combines local feature extraction (CNN) with long-term dependencies (LSTM).

---

4. Bidirectional LSTM

```
model = Sequential([
  TimeDistributed(Dense(projection_dim, activation='linear'), input_shape=input_shape),
  Bidirectional(LSTM(lstm_units, return_sequences=True)),
  Bidirectional(LSTM(lstm_units, return_sequences=False)),
  Dense(dense_units, activation='relu'),
  Dropout(dropout_rate),
  Dense(num_classes, activation='softmax')
])
```

- Uses Bidirectional LSTMs to capture both past and future temporal information.
- Stacking helps model complex patterns.
- Particularly useful in time-series where both directions contain meaningful data.

---

5. Bidirectional LSTM + CNN

```
model = Sequential([
  TimeDistributed(Dense(projection_dim, activation='relu'), input_shape=input_shape),
  Conv1D(filters=64, kernel_size=3, activation='relu'),
  MaxPooling1D(pool_size=2),
  Bidirectional(LSTM(lstm_units, return_sequences=True)),
  Bidirectional(LSTM(lstm_units, return_sequences=False)),
  Dense(dense_units, activation='relu'),
  Dropout(dropout_rate),
```

```
    Dense(num_classes, activation='softmax')
])
```

- Combines the feature extraction strength of CNNs and the bidirectional temporal modeling of LSTMs.
- This hybrid architecture allows the model to learn local trends and global sequences simultaneously.

---

6. Training Setup

All models use:

- **Adam Optimizer** with a tunable learning rate.
- **Categorical Crossentropy Loss** for multi-class classification.
- **EarlyStopping** to halt training when validation loss stops improving.
- **ModelCheckpoint** to save the best performing model.
- **ReduceLROnPlateau** to adapt learning rate during training.

Performance is tracked using:

- **Accuracy**
- **Macro F1-Score** (accounts for class imbalance)
- **Loss Curves** for training and validation

This modular approach allows us to compare architectures and tune hyperparameters (like LSTM units, dropout, learning rate, etc.) to find the best performing setup for the given dataset.

Results

Model Name	LSTM Units	Dense Units	Dropout	LR	Batch	Proj Dim	Accuracy	F1 Score
default	128	128	0.2	0.005	64	32	0.506	0.519
lstm_256	256	128	0.2	0.005	64	32	0.533	0.542

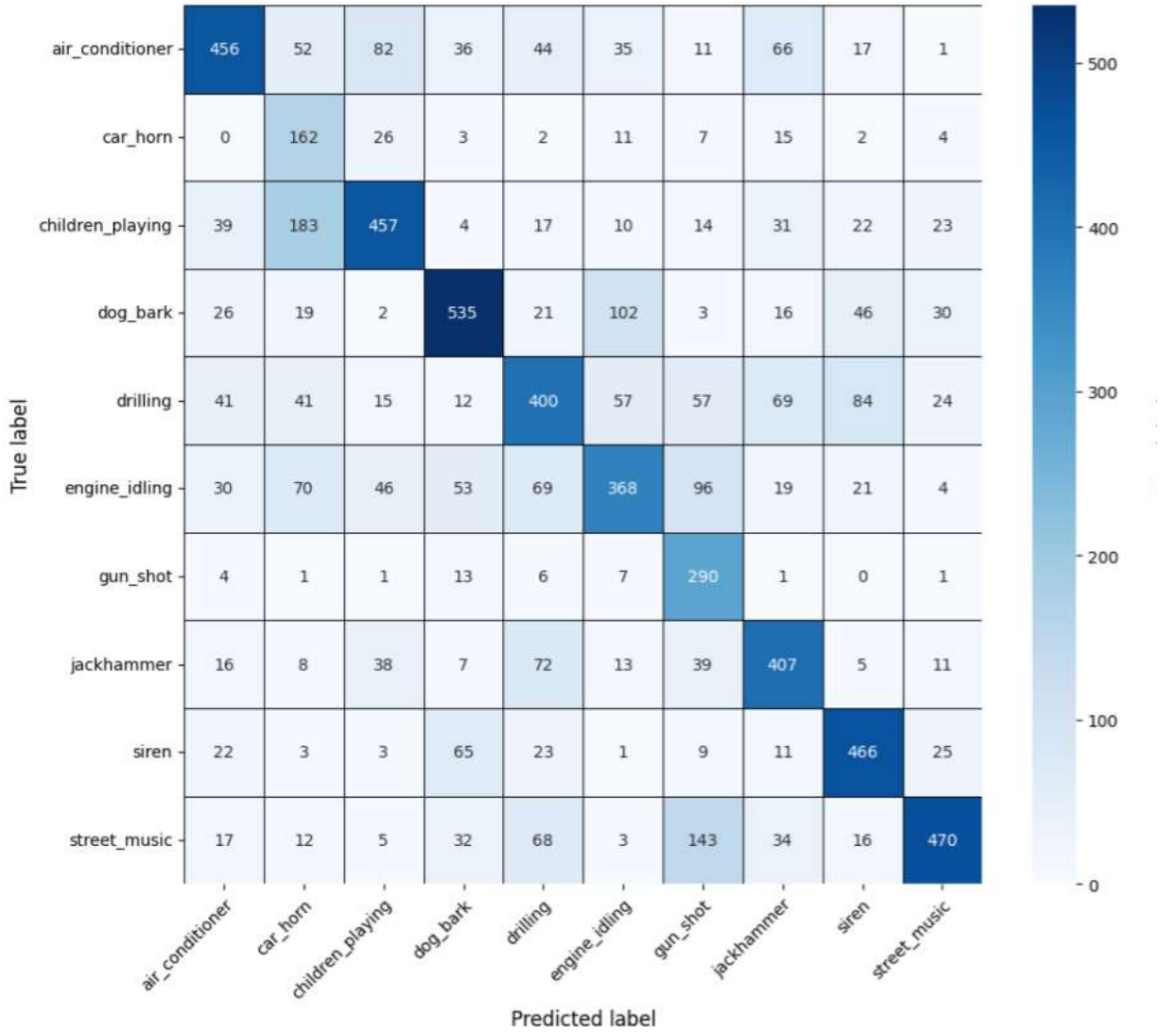
Model Name	LSTM Units	Dense Units	Dropout	LR	Batch	Proj Dim	Accuracy	F1 Score
lstm_64	64	128	0.2	0.005	64	32	0.555	0.546
dropout_0.3	128	128	0.3	0.005	64	32	0.506	0.502
dropout_0.1	128	128	0.1	0.005	64	32	0.485	0.498
dropout_0.4	128	128	0.4	0.005	64	32	0.536	0.544
lr_05	128	128	0.2	0.05	64	32	0.094	0.017
lr_001	128	128	0.2	0.001	64	32	0.432	0.405
lr_01	128	128	0.2	0.01	64	32	0.574	0.581
batch_32	128	128	0.2	0.005	32	32	0.535	0.530
batch_128	128	128	0.2	0.005	128	32	0.468	0.428
new_default	128	128	0.5	0.001	64	32	0.533	0.543
proj_64	128	128	0.5	0.001	64	64	0.542	0.551
proj_16	128	128	0.5	0.001	64	16	0.528	0.540
proj_8	128	128	0.5	0.001	64	8	0.471	0.476
proj_4	128	128	0.5	0.001	64	4	0.464	0.477
best_individuals_combined	64	64	0.4	0.01	64	16	0.485	0.461
default_lstm_with_54_features	128	256	0.5	0.001	64	64	0.498	0.466
default_lstm_cnn	128	256	0.5	0.001	64	64	0.473	0.407
default_bidirectional_lstm	128	256	0.5	0.001	64	64	0.553	0.573
default_bidirectional_lstm_cnn	128	256	0.5	0.001	64	64	0.600	0.618
bidirectional_lstm_cnn_lstm_64_dense_128_dropout_06	64	128	0.6	0.001	64	64	0.552	0.579
bidirectional_lstm_cnn_lstm_32_dense_64_dropout_07_lr_0001	32	64	0.7	0.0001	64	64	0.566	0.580
bidirectional_lstm_cnn_lstm_64_dense_128_dropout_07_lr_0001	64	128	0.7	0.0001	64	64	0.545	0.562

Analysis

Top 3 Models

Model Name	Accuracy	F1 Score
default_bidirectional_lstm_cnn	**0.600**	**0.618**
bidirectional_lstm_cnn_lstm_64_dense_128_dropout_06	0.552	0.579
bidirectional_lstm_cnn_lstm_32_dense_64_dropout_07_lr_0001	0.566	0.580

default_bidirectional_lstm_cnn is the **best model overall** (highest F1 and accuracy).



air_conditioner was predicted wrong 344 times

car_horn was predicted wrong 70 times

children_playing was predicted wrong 343 times

dog_bark was predicted wrong 265 times

drilling was predicted wrong 400 times

engine_idling was predicted wrong 408 times

gun_shot was predicted wrong 34 times

jackhammer was predicted wrong 209 times

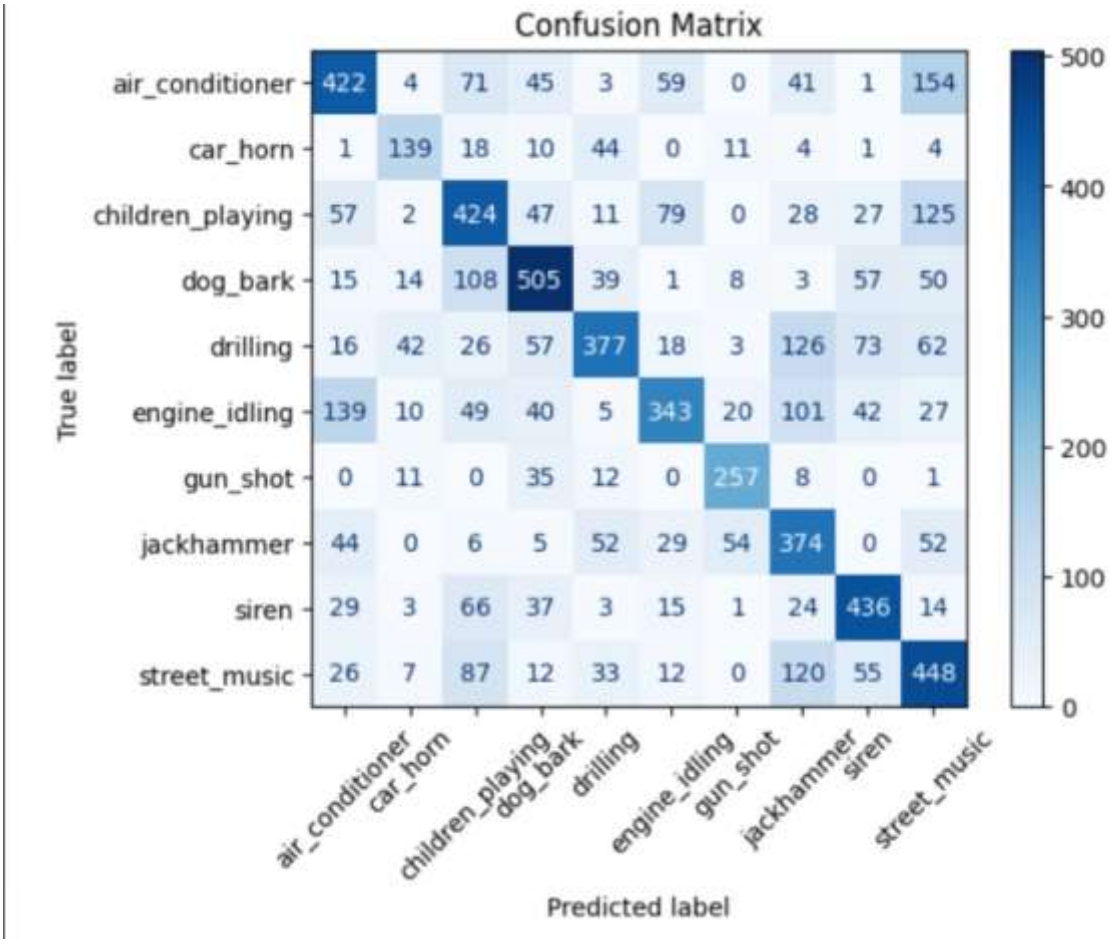
siren was predicted wrong 162 times

street_music was predicted wrong 330 times

Most Confusing Class was engine_idling

Best Single-LSTM Model (no CNN or bidirectional)

Model Name	Accuracy	F1 Score
lr_01	**0.574**	**0.581**



air_conditioner was predicted wrong 378 times

car_horn was predicted wrong 93 times

children_playing was predicted wrong 376 times

dog_bark was predicted wrong 295 times

drilling was predicted wrong 423 times

engine_idling was predicted wrong 433 times

gun_shot was predicted wrong 67 times

jackhammer was predicted wrong 242 times

siren was predicted wrong 192 times

street_music was predicted wrong 352 times

Most Confusing Class was engine_idling and it was most confused with air_conditioner (139 times) and jackhammer (101 times)

Transformer Model

Architecture:

🔷 1. Input Projection

The model starts by projecting each input timestep (of size 54) into a higher-dimensional space using a linear layer. This transforms the raw input into a format suitable for attention mechanisms, with dimensionality d_model.

🔷 2. Positional Encoding

Since Transformer models lack inherent knowledge of sequence order, sinusoidal positional encodings are added to the input embeddings. These encodings allow the model to distinguish positions in the sequence and capture relative timing information.

🔷 3. Encoder Blocks

The core of the architecture consists of N identical **encoder blocks**, each composed of:

- **Multi-Head Self-Attention**: Enables the model to attend to different positions in the sequence simultaneously.
- **Feed-Forward Network**: A two-layer MLP applied independently to each position.
- **Residual Connections & Layer Normalization**: Stabilize and accelerate training by preserving gradients and normalizing layer outputs.

🔷 4. Multi-Head Attention

Each attention block splits the input into multiple heads. Each head learns different types of relationships between positions. The scaled dot-product attention mechanism is used within each head.

🔷 5. Feedforward Network

After attention, a position-wise feedforward network processes each timestep independently to enhance representation learning.

🔷 6. Output Layer

The final encoder output (shape: batch × sequence × d_model) is **mean-pooled across the sequence** to create a fixed-size vector per input sample. This vector is then passed to a final linear classifier for prediction across 10 classes.

🔷 Key Hyperparameters

- seq_len: Length of input sequence.
- d_model: Dimensionality of projected features.
- nhead: Number of attention heads.
- d_ff: Hidden size of the feedforward network.
- N: Number of repeated encoder blocks.

Transformation

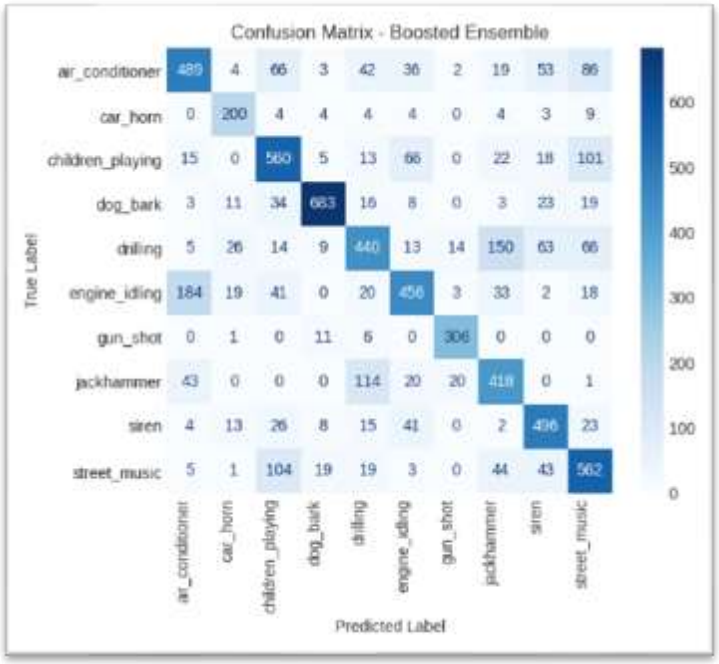
🔷 Extreme values are limited by clipping features to between the 5th and 95th percentiles to reduce outlier effects.

🔷 A RobustScaler is fitted on the clipped training data to scale features based on median and interquartile range, making it less sensitive to outliers.

Techniques

🔷 **Adaboosting** for well performed 10 models such that the final probability for each class is the weighted average of all models based on accuracy

Metrics



Most Confusing Class: “Engine_idling**”**