Aim

To develop a neural network model for classifying urban sounds using the UrbanSound8K dataset.

Objective

1. Extract meaningful features from audio files using Mel-Frequency Cepstral Coefficients (MFCC).
2. Train a neural network model to classify different urban sound categories.
3. Evaluate the model's performance and predict the class of new audio samples.

Summary

This project focuses on classifying urban sounds using the UrbanSound8K dataset. The process involves extracting features from audio files using MFCC, training a neural network model, and evaluating its performance. The project aims to accurately classify various urban sound categories, enhancing applications in sound recognition and environment understanding.

Tools and Libraries Used

- **pandas**: For data manipulation and analysis.
- **librosa**: For audio processing and feature extraction.
- **numpy**: For numerical computations.
- **tqdm**: For progress bar implementation.
- **scikit-learn (sklearn)**: For preprocessing and train-test splitting.
- **TensorFlow**: For building and training the neural network model.

Procedure

1. **Load Metadata and Audio Files**:
    - Load the metadata file using pandas.
    - Define the path to the audio dataset.

CODE:

```
audio_dataset_path='UrbanSound8K/audio/'

metadata=pd.read_csv('UrbanSound8K/metadata/UrbanSound8K.csv')

metadata.head()
```

2. **Feature Extraction**:
    - Define a function to extract MFCC features from audio files.

CODE:

```
def features_extractor(file):
   audio, sample_rate = librosa.load(file_name, res_type='kaiser_fast')

   mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)

   mfccs_scaled_features = np.mean(mfccs_features.T, axis=0)

   return mfccs_scaled_features
```

3. **Iterate and Extract Features**:
    - Iterate through each audio file in the dataset and extract features using the defined function.
    - Store the extracted features and corresponding class labels.

CODE:

```
extracted_features=[]

for index_num, row in tqdm(metadata.iterrows()):

   file_name = os.path.join(os.path.abspath(audio_dataset_path), 'fold'+str(row["fold"])+'/',
str(row["slice_file_name"]))

   final_class_labels = row["class"]

   data = features_extractor(file_name)

   extracted_features.append([data, final_class_labels])

extracted_features_df = pd.DataFrame(extracted_features, columns=['feature', 'class'])
```

4. **Prepare Data for Training**:
    - Convert the features and labels into numpy arrays.
    - Encode the labels using one-hot encoding.

CODE:

```
X = np.array(extracted_features_df['feature'].tolist())

y = np.array(extracted_features_df['class'].tolist())

from tensorflow.keras.utils import to_categorical

from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()

y = to_categorical(labelencoder.fit_transform(y))
```

5. **Train-Test Split**:
    - Split the data into training and testing sets.

CODE:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

6. **Build and Train Neural Network Model**:
   - Define a Sequential model with Dense, Activation, and Dropout layers.
   - Compile the model and train it using the training data.

CODE:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation
num_labels = y.shape[1]
model = Sequential()
model.add(Dense(100, input_shape=(40,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(100))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')

from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime
num_epochs = 100
num_batch_size = 32
```

```
checkpointer = ModelCheckpoint(filepath='saved_models/audio_classification.hdf5', verbose=1,
save_best_only=True)

start = datetime.now()

model.fit(X_train, y_train, batch_size=num_batch_size, epochs=num_epochs, validation_data=(X_test,
y_test), callbacks=[checkpointer], verbose=1)

duration = datetime.now() - start
```

7. **Evaluate Model**:
    - Evaluate the model's accuracy on the test data.

CODE:

```
test_accuracy = model.evaluate(X_test, y_test, verbose=0)

print(test_accuracy[1])
```

8. **Predict New Audio Data**:
    - Preprocess new audio data and predict its class.

CODE:

```
filename = "UrbanSound8K/drilling_1.wav"

audio, sample_rate = librosa.load(filename, res_type='kaiser_fast')

mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)

mfccs_scaled_features = np.mean(mfccs_features.T, axis=0)

mfccs_scaled_features = mfccs_scaled_features.reshape(1, -1)

predicted_label = model.predict_classes(mfccs_scaled_features)

prediction_class = labelencoder.inverse_transform(predicted_label)

print(prediction_class)
```

Highlights

- **Feature Extraction**: Efficient extraction of MFCC features for robust audio classification.
- **Neural Network Architecture**: Sequential model with multiple dense layers and dropout for regularization.
- **Training and Evaluation**: Use of ModelCheckpoint for saving the best model and evaluation of model performance.
- **Real-Time Prediction**: Capability to preprocess and classify new audio samples.

Conclusion

The project successfully developed a neural network model capable of classifying urban sounds with good accuracy. The model was trained on the UrbanSound8K dataset, using MFCC features for audio representation. The approach demonstrated effective feature extraction, model training, and evaluation techniques, providing a foundation for further improvements and applications in sound recognition systems.