Time Series Classification of Japanese Vowels Using LSTM Neural Network in MATLAB

Mehuli Lahiri

***Abstract*—This study implements a Long Short-Term Memory (LSTM) neural network to classify Japanese vowel utterances using MATLAB's `japaneseVowelsTrainData` and `japaneseVowelsTestData`. The LSTM architecture effectively handles variable-length sequences, making it suitable for speech-based time series data. The model achieved a classification accuracy of 95.95%, demonstrating high performance across multiple evaluation metrics including precision, recall, and F1-score.**

***Index Terms*—LSTM, Japanese Vowels, Time Series Classification, Neural Network, MATLAB, Speech Recognition.**

I. Introduction

Time series classification is an essential task in fields such as speech recognition and medical diagnostics. Unlike static feature vectors, time series data are sequential and variable in length, requiring architectures capable of preserving temporal dependencies. This work uses MATLAB's `japaneseVowelsTrainData`, a benchmark dataset for speaker-dependent speech classification, and applies a Long Short-Term Memory (LSTM) network to learn temporal dynamics in vowel sequences.

II. Dataset Overview

The dataset contains time series representing vowel utterances from different speakers. Each observation is a matrix where rows represent features (MFCC-like) and columns represent time steps. The task is to classify each sequence into one of nine classes.

III. Methodology

A. Preprocessing

The data is provided as cell arrays suitable for direct input into MATLAB's sequence-to-label LSTM structure. No normalization or reshaping was required.

B. Network Architecture

The LSTM network architecture consists of:

- A sequence input layer with input size 12.
- An LSTM layer with 100 hidden units and output mode set to 'last'.
- A fully connected layer with 9 output classes.
- A softmax layer.
- A classification layer.

C. Training Parameters

The model was trained using the Adam optimizer with the following configuration:

- Epochs: 100
- Mini-batch size: 16
- Sequence length handling: Longest

IV. Training Process

Figure 1 shows the training accuracy and loss over 100 epochs.

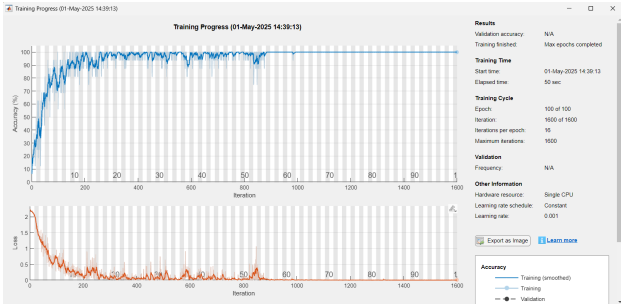

Fig. 1: Training progress (accuracy and loss over epochs)

V. Evaluation Metrics

The model achieved an overall classification accuracy of 95.95%. Figure 2 shows the confusion matrix for the test dataset.

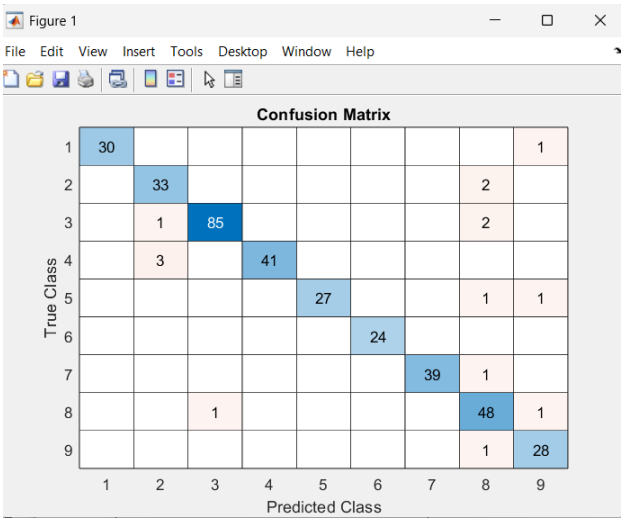

Fig. 2: Confusion matrix of LSTM predictions

Precision, recall, and F1-score for each class are summarized in Figure 3.

```
    predicted = YPred == classes(i);

    tp = sum(actual & predicted);
    fp = sum(~actual & predicted);
    fn = sum(actual & ~predicted);

    precision(i) = tp / (tp + fp + eps);
    recall(i) = tp / (tp + fn + eps);
    f1(i) = 2 * (precision(i)*recall(i)) / (precision(i)+recall(i) + eps);
end

table(classes, precision, recall, f1)
Accuracy: 0.95946

ans =

  9×4 table

    classes    precision    recall      f1
    _______    _________    ______    _______

     {'1'}           1      0.96774    0.98361
     {'2'}     0.89189      0.94286    0.91667
     {'3'}     0.98837      0.96591    0.97701
     {'4'}           1      0.93182    0.96471
     {'5'}           1      0.93103    0.96429
     {'6'}           1            1          1
     {'7'}           1        0.975    0.98734
     {'8'}     0.87273         0.96    0.91429
     {'9'}     0.90323      0.96552    0.93333
```

Fig. 3: Precision, recall, and F1-score per class

VI. Implementation Code

The model was implemented in MATLAB using the Deep Learning Toolbox. Figure 4 shows the main section of the LSTM implementation.

```
>> [XTrain, YTrain] = japaneseVowelsTrainData;
[XTest, YTest] = japaneseVowelsTestData;
>> numObservations = numel(XTrain);
sequenceLength = cellfun(@(x) size(x,2), XTrain);
classes = categories(YTrain);
numClasses = numel(classes);
>> inputSize = size(XTrain{1},1);
numHiddenUnits = 100;

layers = [
    sequenceInputLayer(inputSize)
    lstmLayer(numHiddenUnits,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];
>> options = trainingOptions('adam', ...
    'MaxEpochs',100, ...
    'MiniBatchSize', 16, ...
    'SequenceLength','longest', ...
    'Shuffle','every-epoch', ...
    'Plots','training-progress', ...
    'Verbose',false);
>> options = trainingOptions('adam', ...
    'MaxEpochs',100, ...
    'MiniBatchSize', 16, ...
    'SequenceLength','longest', ...
    'Shuffle','every-epoch', ...
    'Plots','training-progress', ...
    'Verbose',false);
>> net = trainNetwork(XTrain, YTrain, layers, options);
YPred = classify(net, XTest, 'SequenceLength','longest');
accuracy = sum(YPred == YTest) / numel(YTest);
```

Fig. 4: LSTM network training code in MATLAB

VII. Results and Discussion

The model performs well across all classes, with precision and recall exceeding 90% in nearly all cases. Misclassifications primarily occurred between classes ‘2’, ‘8’, and ‘9’, which may be due to similarities in speech acoustics.

VIII. Conclusion

This work demonstrates the effectiveness of LSTM networks in classifying time series speech data. The high accuracy and balanced evaluation metrics indicate that the model generalizes well and can be extended to other speech recognition tasks.