# Real-Time Emotion Recognition Using Liquid Neural Networks

**Soham Vijaykumar Faldu sf4117,**
**Manav Parikh mpp6923***,
**Ashutosh Kumar ak10514***
New York University

**Abstract**

Understanding human emotions in real-time can be crucial for enhancing applications like virtual mental health therapy tools, customer service bots, and virtual assistants. Liquid Neural Networks (LNNs), with their dynamic adaptability to time-varying data, provide a solution for accurately recognizing and responding to real-time emotional changes.

## Literature Review

### Baseline Paper: Etienne et al. (2018)

Etienne et al. (2018) introduced a CNN+LSTM architecture for speech emotion recognition (SER) using the IEMOCAP dataset. The model combined convolutional layers for feature extraction with Bi-LSTM layers for long-term dependency aggregation. To address class imbalance and data scarcity, the authors employed vocal tract length perturbation (VTLP) for data augmentation, achieving competitive results with weighted and unweighted accuracies of 64.5% and 61.7%, respectively. The study emphasized the importance of 10-fold cross-validation for SER evaluation, providing a strong baseline for subsequent research.

### Huang et al. (2023)

Huang et al. (2023) proposed a real-time SER framework integrating dynamic feature extraction and lightweight neural networks. The dynamic module adapted to temporal variations in speech, enhancing emotional nuance capture, while its efficient model design ensured computational feasibility. It was evaluated on benchmark datasets and the approach outperformed traditional methods, demonstrating versatility for multilingual and cross-cultural applications, such as virtual assistants and mental health monitoring.

### Multilingual Speech Emotion Recognition (2022)

The 2022 study on Multilingual Speech Emotion Recognition (M-SER) addressed variability in linguistic and prosodic features across languages. By combining traditional acoustic features with language-specific embeddings, the authors improved recognition accuracy on diverse datasets. The work highlighted cultural factors influencing emotional expressions, enhancing the inclusivity of SER systems for global applications like multilingual customer service and mental health support.

### Hasani et al. (2021)

Hasani et al. (2021) explored Liquid Neural Networks (LNNs) for real-time time-series data. LNNs, inspired by liquid state machines, outperformed traditional architectures like RNNs and LSTMs in dynamic environments due to their continuous-time processing and adaptability to sudden changes. Their robustness and efficiency make LNNs suitable for applications such as autonomous driving and healthcare monitoring.

## Dataset

The IEMOCAP dataset, collected at the University of Southern California (USC), consists of 12 hours of audio and video recordings from 10 professional actors (5 women and 5 men) in 5 sessions of dyadic interactions. The dataset includes both scripted and spontaneous (improvised) dialogues. Each utterance is annotated with emotion labels by USC students, with final labels determined by majority vote.

The distribution of these emotions in the dataset is as follows:

- Neutral $\approx 40\%$
- Sad $\approx 25\%$
- Angry $\approx 20\%$
- Happy $\approx 15\%$

## Data Preprocessing

The preprocessing of data is a critical part of the model's performance. The audio data undergoes several transformations to extract meaningful features. First, Mel-Frequency Cepstral Coefficients (MFCC), Delta MFCC (first-order derivatives), and Delta-Delta MFCC (second-order derivatives) are computed. These features are concatenated to form a feature vector of shape (time_steps, 39), where time_steps represents the number of time steps in the sequence and features represents the number of features per time step (39 in this case).

*These authors contributed equally.

To ensure uniformity, all sequences are padded to a maximum length. Data augmentation techniques, including pitch shifting, time stretching, and adding random noise, are applied to enhance the model's generalization capabilities. The features are then standardized using StandardScaler to ensure a mean of 0 and a standard deviation of 1. To address class imbalance, class weights are computed using compute_class_weight, which balances the contribution of each class during training. Finally, the target labels are one-hot encoded to facilitate the use of categorical cross-entropy as the loss function.

## Model Architecture

The model architecture for our project is specifically designed to address the challenges of real-time emotion recognition by leveraging a custom implementation of Liquid Neural Networks (LNN). The core innovation lies in the development and integration of a custom RNN cell—the LTCCell—which is inspired by the dynamics of ordinary differential equations (ODEs). This custom implementation ensures both computational efficiency and adaptability, making it highly suitable for sequence modeling tasks.

### Core Components of LTCCell

The LTCCell is the main part of our architecture, designed to effectively handle temporal dependencies in sequential data. It incorporates 16 units, a configuration selected after rigorous experimentation to create a balance between model complexity and performance. L2 regularization with a coefficient of 0.001 is applied to mitigate overfitting, ensuring robust generalization across diverse inputs. Furthermore, the cell leverages custom activation functions to stabilize the training process: a clipped ReLU function constrains outputs to a maximum value of 20.0, while a clipped sigmoid function scales outputs to a maximum of 10.0 for controlled gradient behavior.

To model temporal dynamics, the LTCCell employs an ODE-like iterative process. Using a time step ($\Delta t$) of 0.01, the cell updates its state iteratively over multiple steps, simulating the continuous evolution of the system. This approach captures complex temporal relationships in the data, making the architecture highly effective for tasks requiring real-time adaptability.

### Integration into the Model

The LTCCell is integrated into an RNN layer configured to output only the final state. This final state is then passed through a dense layer with four output units, each representing an emotion class: happy, angry, neutral, and sad. A softmax activation function is applied to produce a probability distribution over the classes. To further enhance generalization, L2 regularization with a coefficient of 0.001 is also applied to the dense layer.

## Mathematical Formulation of LTCCell

The core of the LTCCell's operation is governed by the following state update equation:

$$\frac{dh}{dt} = \frac{-h_{\text{prev}} + \tanh(W \cdot [x, h_{\text{prev}}] + b)}{\text{softplus}(\tau)}$$

Here, $h_{\text{prev}}$ represents the previous state, $W$ is the trainable weight matrix, $b$ is the bias vector, and $\tau$ is a trainable time constant. The use of softplus($\tau$) ensures numerical stability by maintaining positivity in the denominator. The state is updated iteratively over $N$ steps using the formula:

$$h_{\text{prev}}^{(t+1)} = h_{\text{prev}}^{(t)} + \Delta t \cdot \frac{dh}{dt}$$

After completing the iterative updates, custom activation functions are applied to refine the state:

**Clipped ReLU Activation** This activation stabilizes the model by constraining the state to a maximum value of 20.0:

$$h = \min(\max(0, h), 20.0)$$

**Clipped Sigmoid Scaling** Further modulates the state by applying scaling and capping mechanisms:

$$h = h \cdot \frac{\text{sigmoid}(C)}{\text{sigmoid}(G)}$$

Here, $C$ and $G$ are trainable parameters that control scaling and modulation.

## Loss Function and Hyperparameters

The model is trained using the categorical cross-entropy loss function, which is appropriate for multi-class classification tasks with one-hot encoded labels. The Adam optimizer is used with a learning rate of 0.0005. The training process involves the following hyperparameters:

- Batch Size: 64
- Number of Epochs: 50
- Early Stopping:
  - Patience: 5 epochs
  - Restore Best Weights: True
- Learning Rate Reduction:
  - Factor: 0.5
  - Patience: 2 epochs

These hyperparameters were selected based on empirical evaluation and tuning. Early stopping is applied to prevent overfitting, while the learning rate is reduced if the validation loss does not improve for 2 consecutive epochs.

## Training and Evaluation

The model was trained on a dataset consisting of 14,368 audio samples, with a validation set of 3,592 samples. The training process involved 50 epochs, during which the model achieved a training accuracy of 64.99% and a validation accuracy of 63%. The training loss decreased to 0.9118, while the validation loss stabilized at 0.9308.

The model was evaluated on a separate test set, achieving a test accuracy of 63%. The classification report provides a detailed breakdown of the model's performance:

- Angry: Precision: 0.79, Recall: 0.76, F1-score: 0.77
- Happy: Precision: 0.00, Recall: 0.00, F1-score: 0.00
- Sad: Precision: 0.60, Recall: 0.73, F1-score: 0.66
- Neutral: Precision: 0.57, Recall: 0.70, F1-score: 0.63

The model performs well on the "angry," "sad," and "neutral" classes but struggles with the "happy" class, which has a precision and recall of 0.00. This indicates that further improvements are needed, particularly in handling the "happy" class.

## Real-Time Implementation

The implementation uses the pre-trained LNN model with the best accuracy score. The process begins with preprocessing the audio data to extract meaningful features. The `extract_features_from_audio` function computes Mel-Frequency Cepstral Coefficients (MFCC), Delta MFCC, and Delta-Delta MFCC, which are concatenated and transformed into a feature vector. These features are then padded or truncated to match the model's expected input shape, ensuring consistency. The audio file is split into 5-second segments to handle longer recordings, and each segment is processed independently. For each segment, the features are extracted, reshaped to include a batch dimension, and fed into the model for emotion prediction. The model outputs a probability distribution over the emotion classes, and the class with the highest probability is selected as the detected emotion.

To evaluate the system's efficiency, latency is measured for each segment by calculating the time taken to process the segment and make a prediction. This allows for an assessment of real-time performance, which is critical for applications requiring quick response times. The detected emotion is mapped to a corresponding label (e.g., angry, happy, sad, neutral) based on the model's output indices. The implementation ensures flexibility by handling variable-length audio files and providing interpretable results. Overall, the approach focuses on preprocessing audio data, segmenting it for efficient processing, and using the pre-trained LNN model to classify emotions with measurable latency. The implementation can be used in two ways, either in real time or by uploading an audio file.

## Results

We began with a model configuration of 128 units, resulting in approximately 21,000 parameters. However, the training process proved unstable, with accuracy plateauing below 30% and the loss curve exhibiting significant oscillations. To address these issues, we introduced an L2 regularization term. While this helped stabilize the training curve to some extent, the oscillations persisted, and the improvement in accuracy was marginal.

Contrary to expectations, increasing the number of units in the LNN led to a steady decline in accuracy. Whereas, reducing the number of units significantly improved performance. The best results were achieved with a lightweight configuration of 16 units and just 710 parameters, where the model reached a maximum accuracy of 63.03%. This configuration also demonstrated exceptional efficiency, with a latency of just 0.1 seconds per prediction, making it highly suitable for real-time applications.

We further experimented with structural modifications, such as double-stacking two Liquid Time-Constant (LTC) cells and introducing a dropout layer. However, these changes reduced the accuracy to 25%, highlighting the effectiveness of simpler architectures for this task. A summary of some of these results is presented below in Table 1.

## Discussion

The results demonstrate that reducing model complexity led to better performance in the emotion recognition task. This observation aligns with the LNN's design philosophy of dynamic adaptability with fewer parameters, making it particularly suitable for tasks with temporal dependencies and real-time requirements.

1. Increasing the number of units introduced excessive flexibility, which likely caused overfitting to the training data and degraded generalization. Simplifying the model reduced overfitting and enhanced performance.
2. While L2 regularization helped marginally, it was insufficient to address instability in larger models. Adding structural complexity, such as stacked LTC cells and dropout layers, negatively impacted performance.
3. The minimal latency of the 16-unit model highlights its practicality for real-time emotion recognition. The efficiency gains make this configuration ideal for deployment in applications like virtual mental health therapy and real-time customer service.

These findings suggest that for tasks like emotion recognition, the inherent adaptability of LNNs enables simpler models to outperform more complex counterparts. This not only reduces computational overhead but also aligns with the goal of minimizing latency, a critical metric for real-time applications.

## Conclusion

We aimed to surpass the baseline accuracy of 65% for emotion recognition on the IEMOCAP dataset. While our model achieved an accuracy of 63%, slightly below the baseline, this result highlights that simpler models, such as Random Forests, can sometimes outperform neural networks in terms of raw accuracy. The custom Liquid Neural Network (LNN) model we developed is exceptionally lightweight, utilizing only 710 parameters to achieve comparable accuracy on the same dataset. This reduction in complexity compared to the baseline models highlights the efficiency of LNNs. Additionally, the model's latency of just 0.1 seconds per prediction makes it highly suitable for deployment in fast-paced, real-time environments.

In conclusion, while accuracy remains an important metric, our work highlights the importance of balancing performance with efficiency and usability. The lightweight nature and low latency of our model make it a compelling choice for real-world applications, particularly in domains where computational resources and response times are critical.

| Learning Rate | Reg Rate | Batch Size | Units | Epochs | LR Reduction Factor | Early Stopping Patience | Over sampling | Accuracy |
|---|---|---|---|---|---|---|---|---|
| 0.0005 | 0.0005 | 64 | 16 | 50 | 0.5 | 5 | FALSE | 63.00% |
| 0.0005 | 0.0005 | 24 | 16 | 100 | 0.5 | 5 | FALSE | 62.50% |
| 0.0005 | 0.0005 | 64 | 16 | 100 | - | 10 | FALSE | 61.44% |
| 0.0001 | 0.0005 | 64 | 8 | 50 | - | 10 | TRUE | 62.84% |
| 0.001 | 0.0001 | 32 | 32 | 100 | 0.5 | 10 | FALSE | 24.75% |
| 0.0001 | 0.0001 | 64 | 12 | 50 | - | 5 | FALSE | 60.20% |
| 0.001 | 0.001 | 128 | 24 | 100 | - | 5 | TRUE | 57.20% |
| 0.001 | 0.0005 | 64 | 24 | 100 | - | 5 | TRUE | 55.34% |
| 0.0005 | 0.0005 | 32 | 12 | 100 | 0.5 | 5 | FALSE | 61.50% |

Table 1: Summary of Model Configurations and Results

## References


1. Etienne, C., Fidanza, G., Petrovskii, A., Devillers, L., & Schmauch, B. (2018). CNN+LSTM Architecture for Speech Emotion Recognition with Data Augmentation.
2. Huang et al., "EmotionNAS: Two-stream Neural Architecture Search for Speech Emotion Recognition," 2023.
3. Multilingual Speech Emotion Recognition with Multi-Gating Mechanism and Neural Architecture Search, 2022.
4. Hasani et al., "Liquid Neural Networks for Adaptive Sequential Learning," 2021.
5. C. Busso, M. Bulut, C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. Chang, S. Lee, and S. Narayanan, "IEMOCAP: Interactive emotional dyadic motion capture database," Journal of Language Resources and Evaluation, vol. 42, no. 4, pp. 335-359, December 2008.


## Code

You can access the code in this Github Repository

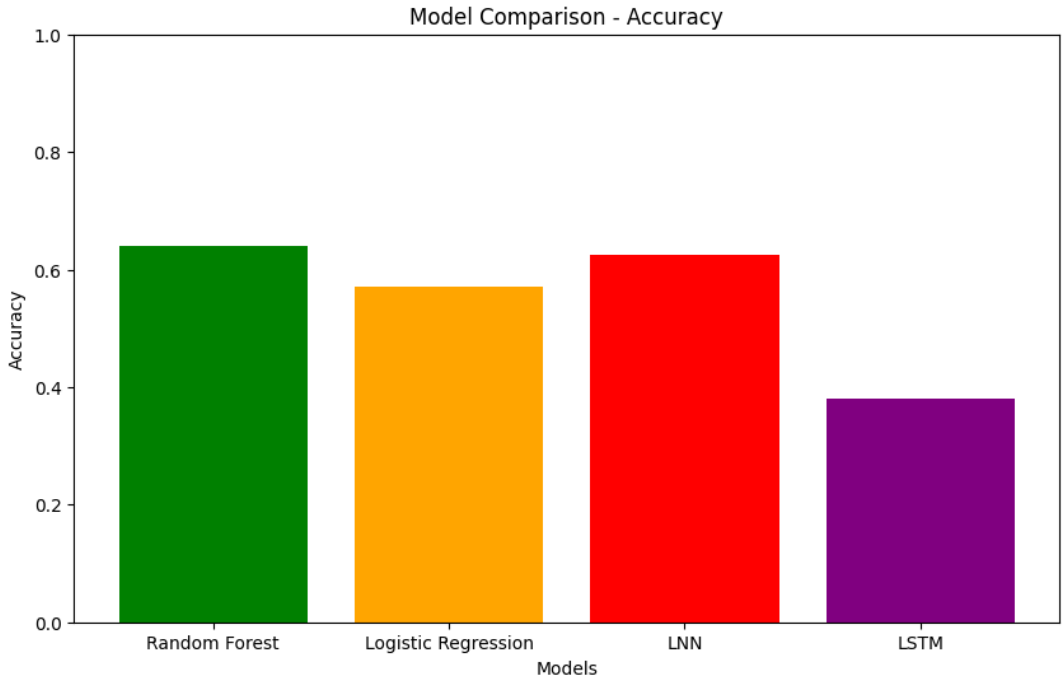


Figure 1: Training and Validation Accuracy and Loss Over Epochs

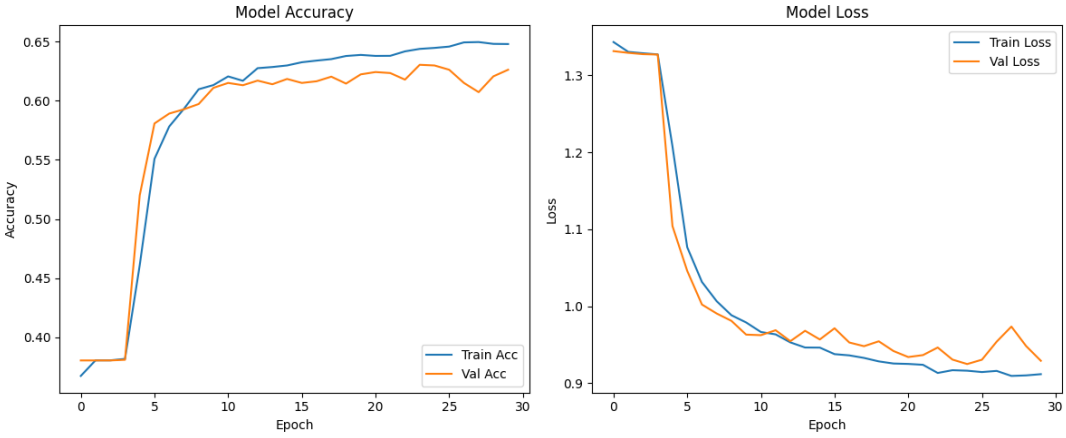


Figure 2: Comparison of Accuracy Scores for Different Models