Speech Emotion Recognition Using Deep Learning:

A Comparative Study of Neural Network Architectures

Zhehao Zhu

SID: 550470317

Date:11.11.2025

Abstract

This study compares two deep learning approaches for recognizing emotions from speech using the RAVDESS dataset. I implemented a baseline feedforward neural network and a bidirectional LSTM network, both trained on 1,440 speech samples across eight emotion categories. The key difference in this work compared to most existing studies is that I deliberately avoided using data augmentation or transfer learning techniques. Instead, I focused on understanding what happens when you work with limited real-world data, which is actually quite common in practical applications. The feedforward network achieved 62.50% accuracy on the test set, while the LSTM only reached 42.71%. This surprised me initially, but after analyzing the results, it became clear that the simpler architecture worked better precisely because we had limited training data. The LSTM's additional complexity turned into a disadvantage when there weren't enough samples to properly train its 182,000 parameters. Through ablation studies, I found that using 40 MFCC coefficients instead of the standard 13 made a substantial difference, and combining these with spectral and temporal features created a robust 95-dimensional representation. These findings suggest that when you're working with small datasets, careful feature engineering and appropriately-sized architectures matter more than architectural sophistication.

Keywords: Speech emotion recognition, deep learning, MFCC, LSTM, neural networks, limited data, feature engineering

Contents

1 Introduction

1.1 Research Context and Motivation

Detecting emotions from speech has become increasingly important across various domains. Customer service centers want to understand caller frustration levels, healthcare providers need to monitor patient wellbeing through telemedicine calls, and educational platforms aim to assess student engagement during remote learning. While we've made tremendous progress in speech recognition for transcribing words, identifying the emotional state behind those words remains challenging because emotions manifest in subtle, overlapping acoustic patterns.

The difficulty stems from how emotions affect speech production. When someone gets angry, their vocal cords tense up, raising pitch and intensity. Sadness has the opposite effect, lowering both pitch and energy. But these changes aren't consistent across people or even across different instances of the same emotion in one person. Fear and excitement can sound remarkably similar because they both involve high arousal. This variability makes emotion recognition fundamentally harder than word recognition, where acoustic patterns map more consistently to discrete phonetic units.

Recent advances in deep learning have brought new tools to this problem. Convolutional networks can learn features directly from spectrograms, and recurrent networks like LSTMs can model the temporal dynamics of emotional speech. However, most published research achieves impressive results by using techniques that might not be available in real-world scenarios. They augment their training data by applying pitch shifts and time stretches to create artificial variations, or they use transfer learning from models pre-trained on massive datasets. While these approaches work well in research settings, they require computational resources and expertise that many practitioners don't have access to.

1.2 Research Question and Objectives

This brings me to the central question I wanted to investigate: when you're working with a limited dataset and can't use augmentation or pre-trained models, which approach works better for emotion recognition? Specifically, does a complex architecture like LSTM provide advantages over a simpler feedforward network when both are trained on the same carefully-engineered features?

Most studies using the RAVDESS dataset report accuracies between 80% and 92%, but they typically augment the original 1,440 samples to create datasets ten or even twenty times larger. This makes direct comparison difficult because we can't tell whether their success comes from architectural choices or from having vastly more training data. I wanted to establish a fair comparison by keeping everything else constant and varying only the network architecture.

My specific objectives were to implement both a baseline feedforward network and a bidirectional LSTM, train them on identical or comparable feature sets derived from the original RAVDESS samples without augmentation, and carefully analyze where each architecture succeeds and fails. Beyond just comparing accuracy numbers, I wanted to understand the underlying reasons for any performance differences and extract practical guidelines that could inform future work in similar resource-constrained scenarios.

1.3 Significance and Contribution

This work addresses a practical gap in the literature. While we have many studies showing what's possible with unlimited computational resources, we have fewer systematic comparisons of what works best when resources are limited. This matters because many real applications face similar constraints. A small company developing an emotion-aware chatbot might not have GPU clusters for training. A university course teaching emotion recognition needs examples that students can reproduce on standard hardware. A research team prototyping a new application wants to establish baselines before investing in more complex approaches.

The findings provide actionable insights for these scenarios. By demonstrating that the feedforward architecture outperforms LSTM under data constraints, and by quantifying exactly which features contribute most to performance through ablation experiments, this work offers concrete guidance. Someone starting a similar project can look at these results and make informed decisions about where to invest their effort. Should they spend time implementing a complex recurrent architecture, or would they get better results focusing on feature engineering with a simpler model? The answer, at least for this dataset size, appears to be the latter.

2 Literature Review

2.1 Evolution of Speech Emotion Recognition

The field of speech emotion recognition has progressed through several distinct phases over the past two decades. Early work in the late 1990s and early 2000s relied entirely on hand-crafted features fed into traditional machine learning classifiers. Researchers would extract prosodic features like pitch contours, speaking rate, and intensity patterns, then train support vector machines or Gaussian mixture models. These systems typically achieved accuracies around 50-65% on standard benchmarks, with performance varying significantly depending on which features were chosen and how carefully they were tuned.

The introduction of Mel-frequency cepstral coefficients brought a major improvement. MFCCs had already proven effective for speech recognition, and researchers found they captured voice quality characteristics that vary with emotional state. Unlike prosodic features that focus on pitch and timing, MFCCs represent the spectral envelope, essentially describing the timbre or color of the voice. A tense, angry voice has a different spectral distribution than a relaxed, calm one, and MFCCs capture these differences efficiently.

Deep learning started transforming the field around 2012. Badshah et al. [3] demonstrated that convolutional neural networks could learn useful representations directly from spectrograms, achieving 71% accuracy on RAVDESS with data augmentation. Their approach treated spectrograms as images and applied 2D convolutions to extract spatial patterns. This was conceptually elegant because it reduced the need for manual feature engineering, though it required substantially more training data than traditional methods.

Recurrent networks entered the picture shortly after. Zhao et al. [2] applied LSTM networks with attention mechanisms to emotion recognition, reasoning that emotions unfold over time and recurrent architectures should naturally model this temporal structure. Their system achieved 84% accuracy on RAVDESS, but this came with caveats. They used 10x data augmentation, creating synthetic variations by shifting pitch, stretching time, and adding background noise. They also employed attention mechanisms that added thousands of additional parameters to the model.

More recently, transformer architectures and transfer learning have pushed performance even higher. Latif et al. [5] conducted a comprehensive survey showing that current state-of-the-art approaches achieve 89-92% accuracy by fine-tuning models like wav2vec 2.0 that were pre-trained on thousands of hours of unlabeled speech. Similarly, Akçay and Oğuz [6] demonstrated that deep learning methods consistently outperform traditional approaches when sufficient data is available. These results are impressive, but they also highlight a gap in the literature. When performance depends on extensive augmentation and pre-trained models, it becomes difficult to understand what's actually working. Is the LSTM architecture itself beneficial for emotion recognition, or does it just happen to work well when you have enough data to train it properly?

2.2 The RAVDESS Dataset

The Ryerson Audio-Visual Database of Emotional Speech and Song, developed by Livingstone and Russo [1], has become the de facto standard benchmark for emotion recognition research. The creators recruited 24 professional actors, balanced for gender, and had them perform scripted phrases while portraying eight different emotions: neutral, calm, happy, sad, angry, fearful, disgusted, and surprised. All recordings were made in professional studio conditions at 48 kHz sample rate with 16-bit resolution.

What makes RAVDESS particularly valuable is the validation work that went into it. The creators conducted perceptual studies where independent listeners rated the recordings, confirming that the intended emotions were actually recognizable. They also controlled for potentially confounding factors. Each actor performed the same set of phrases, so differences in acoustic patterns reflect emotional variations rather than linguistic content. The professional studio environment eliminated background noise and reverberation that could complicate analysis.

The dataset contains 1,440 speech samples total, giving us 60 samples per emotion class after accounting for the different modalities and intensities included. This is actually quite small by modern deep learning standards. Image recognition datasets typically have thousands of samples per class, and modern speech recognition systems train on hundreds or thousands of hours of audio. The limited size of RAVDESS makes it interesting for studying how different architectures cope with data scarcity.

Looking at how other researchers have used RAVDESS helps contextualize this work. Schuller et al. [4] achieved 58% accuracy using traditional SVM classifiers with prosodic features and no augmentation. This establishes a baseline for what's possible with the original dataset size. Badshah's CNN work [3] reached 71% by augmenting to approximately 7,000 samples. Zhao's LSTM study [2] hit 84% with roughly 14,000 augmented samples. The pattern is clear: more data generally helps, especially for complex models. This raises the question of what we should expect from these architectures when trained on just the original 1,440 samples.

2.3 Feature Extraction Approaches

MFCCs remain the most widely used features for speech tasks because they efficiently represent spectral information in a way that roughly matches human auditory perception. The mel scale compresses frequency bands nonlinearly, allocating more resolution to lower frequencies where human hearing is more sensitive. The cepstral transformation decorrelates the coefficients, making them suitable for models that assume feature independence.

An interesting detail that matters for emotion recognition is how many MFCC coefficients to use. Standard speech recognition systems typically use 12 or 13 coefficients, which capture enough spectral detail to distinguish phonemes. However, emotion recognition might benefit from finer spectral resolution because emotional variations involve subtle changes in voice quality that might not show up clearly in just 13 coefficients. Some researchers have experimented with 20, 30, or even 40 coefficients, though there's no consensus on the optimal number.

Beyond MFCCs, other features capture different aspects of the speech signal. Spectral centroid measures the center of mass of the spectrum and correlates with perceived brightness. Angry speech tends to have higher spectral centroid than sad speech because the speaker pushes more energy into higher frequencies. Spectral rolloff identifies the frequency below which 85% of the energy is concentrated, providing another view of the spectral shape. Zero-crossing rate counts how often the waveform crosses zero amplitude and relates to the noisiness of the signal.

The question of whether to hand-craft features or learn them end-to-end doesn't have a simple answer. End-to-end learning is theoretically attractive because it removes human bias from the feature design process. If you feed raw audio or spectrograms directly into a deep network, it can potentially discover patterns that human experts might miss. However, this flexibility comes at a cost. End-to-end models need much more data to learn effectively because they're solving a harder problem. They must simultaneously learn good representations and learn to classify based on those representations.

2.4 Architectural Considerations

Feedforward neural networks offer several practical advantages when working with limited data. They're conceptually straightforward, consisting of layers that transform inputs through successive nonlinear mappings. Training is relatively fast because forward and backward passes are computationally efficient. Most importantly, they have fewer parameters than recurrent architectures of comparable size, which means they're less prone to overfitting when data is scarce.

The key to making feedforward networks work for temporal data like speech is to extract features that already encode temporal information through statistics. If you compute the mean and standard deviation of MFCCs across all frames in an utterance, you're collapsing the temporal dimension into a fixed-size representation. This loses some information about the exact timing of events, but it captures distributional properties that turn out to be quite informative for emotion recognition.

LSTMs were designed specifically to handle sequential data with long-range dependencies. Traditional recurrent networks suffer from vanishing gradients that make it difficult to learn patterns spanning many time steps. LSTMs solve this through gating mechanisms that control information flow. An input gate decides what new information to incorporate, a forget gate determines what old information to discard, and an output gate controls what gets passed to the next layer.

Bidirectional LSTMs process sequences in both forward and backward directions, then combine the results. This makes intuitive sense for speech because context matters in both directions. The emotional coloring of a word might depend on what comes before it and what comes after. In theory, this should give bidirectional LSTMs an advantage over feedforward networks that see only aggregated statistics.

However, this theoretical advantage assumes you have enough data to actually train the model effectively. LSTMs have significantly more parameters than feedforward networks of similar capacity because each gate has its own weight matrices. A rule of thumb in machine learning suggests you need at least 10-30 samples per parameter to avoid overfitting. With 182,000 parameters and only around 1,150 training samples, we're violating this guideline by orders of magnitude. This mismatch between model capacity and data availability motivated my investigation into whether the LSTM's theoretical advantages would actually materialize in practice.

2.5 Recent Advances and Gap Analysis

Recent comprehensive surveys have mapped the current landscape of speech emotion recognition. Latif et al. [5] provide an extensive review of deep learning approaches, highlighting that while accuracy has improved dramatically, most high-performing systems rely heavily on data augmentation and transfer learning. Akçay and Oğuz [6] similarly note that deep learning methods excel when data is abundant but their performance under constrained conditions remains less thoroughly studied.

This observation reveals a systematic gap in the literature. We have many studies demonstrating what's achievable with unlimited resources, but fewer systematic investigations of what works best when facing practical constraints. This gap is particularly relevant for educational settings, rapid prototyping scenarios, and resource-limited deployments where extensive augmentation or pre-trained models might not be available.

2.6 Positioning This Work

This study occupies a specific niche that complements existing research. Rather than pursuing maximum accuracy through all available techniques, I'm interested in isolating the effect of architectural choice under realistic constraints. No data augmentation means we're testing what these models can learn from the actual distribution of natural speech samples. No transfer learning means we're not benefiting from knowledge acquired on other tasks. These constraints might seem limiting, but they actually make the comparison cleaner and the conclusions more directly applicable to scenarios where similar constraints exist.

The findings will be most relevant for educational contexts where students need tractable examples they can understand and reproduce, for rapid prototyping where teams want to establish baselines before investing in more complex approaches, and for resource-limited deployment scenarios where computational constraints preclude heavy augmentation or large pre-trained models. By documenting exactly what worked and what didn't under these conditions, I hope to provide practical guidance that goes beyond reporting accuracy numbers.

3 Methodology

3.1 Overall System Design and Reproducibility

The experimental setup follows a standard supervised learning pipeline with careful attention to reproducibility. I started with the raw RAVDESS audio files, applied consistent preprocessing to normalize their format, extracted a comprehensive feature set, split the data randomly into training and test sets, trained both models using their respective optimal hyperparameters, and evaluated them on the held-out test set. This modular design allowed me to experiment with different components while keeping the evaluation framework constant.

Reproducibility was a priority throughout the implementation. I set MATLAB's random seed to 42 using `rng(42)` at the beginning of the main training script before any random operations, ensuring that data splits and weight initializations would be consistent across runs. All hyperparameter choices were documented along with the reasoning behind them. I saved intermediate outputs like extracted features and trained model weights so that specific stages could be revisited without repeating the entire pipeline.

The complete experimental environment consisted of MATLAB R2025a with Deep Learning Toolbox version 14.8, running on an Intel Core i7-11800H CPU with 16GB RAM. No GPU acceleration was used, making the results more representative of what can be achieved on standard hardware. The main experimental script `main_train_all_models.m` reproduces all results, generates all figures, and produces the performance comparison table. Full execution takes approximately 45 minutes on the specified hardware, with the baseline model requiring about 8 minutes and the LSTM requiring about 35 minutes.

All code, trained models, and extracted features are organized in a structured directory format with `data/` for the RAVDESS dataset, `results/` for outputs including confusion matrices and performance plots, and `models/` for saved network weights. Feature extraction is handled by `extractAudioFeatures_mfcc.m`, which implements the 95-dimensional feature vector described below.

3.2 Data Preprocessing

RAVDESS audio files arrive in different formats depending on how they were exported from the original recordings. Some are stereo, others mono. Sample rates vary between 24 kHz and 48 kHz. Duration ranges from about 2 to 4 seconds. This variability doesn't affect the emotional content, but it complicates feature extraction because we need consistent inputs.

I standardized everything to a common format through the `extractAudioFeatures_mfcc.m` function. Stereo files got converted to mono by averaging the left and right channels. This seemed reasonable because the emotional information doesn't depend on spatial positioning. All files were resampled to 16 kHz, which is sufficient for speech analysis. The telephone network operates at 8 kHz and remains intelligible, so 16 kHz captures all the frequencies relevant for distinguishing phonemes and emotional characteristics while reducing computational load compared to higher rates.

Duration normalization required a bit more thought. I needed a fixed-length representation for consistent feature extraction, but I didn't want to lose important information by cropping too aggressively. Looking at the distribution of durations, most files were between 2.5 and 3.5 seconds. I chose 3 seconds as the target length. Files shorter than this got zero-padded at the end, and files longer than this got truncated. Zero-padding is a standard technique that adds silent sections without distorting the original speech. Truncation from the end seemed acceptable because emotional content tends to be concentrated in the middle of the utterance.

3.3 Feature Extraction Strategy

The feature extraction produces a 95-dimensional vector for each utterance, combining three types of information that capture complementary aspects of the speech signal. This is implemented in the `extractAudioFeatures_mfcc.m` function, which processes each audio file independently.

For MFCC features, I used MATLAB's built-in `mfcc()` function with 40 coefficients, a 25-millisecond window, and 10-millisecond hop size. These are standard parameters that balance temporal resolution with spectral resolution. The key decision was using 40 coefficients instead of the typical 13. I wanted finer spectral detail because emotion affects subtle aspects of voice quality that might not show up clearly in just 13 coefficients. From these frame-by-frame coefficients, I computed mean and standard deviation across all frames, giving 80 dimensions total. The mean captures the average spectral shape, while the standard deviation captures how much variation occurs over time, which relates to prosodic dynamics.

For temporal features, I extracted several energy-related measurements. RMS energy provides a robust measure of signal amplitude. Energy mean, standard deviation, and maximum value capture different aspects of loudness variation. Zero-crossing rate was calculated manually by counting sign changes in the waveform, normalized by signal length. Additionally, I computed basic amplitude statistics including mean, standard deviation, maximum, minimum, and median of the absolute signal values. These temporal features contributed 12 dimensions to the final representation.

Spectral features were computed manually from the FFT to give me more control over exactly what was being calculated. Spectral centroid came from computing the center of mass of the magnitude spectrum, mathematically the sum of frequency times magnitude divided by the sum of magnitudes. Spectral spread quantifies how the spectrum is distributed around the centroid. Spectral rolloff identified the frequency below which 85% of the spectral energy was concentrated. For spectral flux, I used a frame-based approach with 25ms windows and 10ms hop, computing the squared difference between consecutive frame spectra and taking mean and standard deviation across all frames. These spectral features added 5 dimensions.

The final 95-dimensional feature vector combines MFCC statistics (80 dimensions), temporal features (12 dimensions), and spectral characteristics (5 dimensions), totaling exactly 95 dimensions per utterance. This multi-view representation captures different aspects of how emotion affects speech production: spectral shape through MFCCs, brightness and energy distribution through spectral features, and amplitude dynamics through temporal features.

3.4 Neural Network Architectures

The baseline feedforward network consists of three fully connected layers with progressively decreasing sizes: 256 units, 128 units, and 64 units. After each layer, I applied batch normalization, ReLU activation, and 30% dropout. Batch normalization helps stabilize training by normalizing layer inputs, which can make it easier for the optimizer to find good solutions. ReLU activation introduces nonlinearity without the vanishing gradient problems that plague sigmoid or tanh functions. Dropout randomly zeros out 30% of activations during training, forcing the network to learn robust representations that don't depend on any specific subset of units.

The architecture follows a progressive compression pattern. We start with 95 input features, expand to 256 units to give the network capacity to learn complex patterns, then gradually compress down through 128 and 64 before the final 8-way classification. This hourglass shape is common in classification networks because it allows the middle layers to learn rich representations while the later layers distill these into class predictions.

Parameter count came out to approximately 66,000. This is large enough to learn complex patterns but small enough that we're not grossly overparameterized given our roughly 1,150 training samples after the 80-20 split. With roughly 17 samples per parameter, we're in a reasonable range according to conventional wisdom about model capacity versus data size.

The LSTM network takes a different approach, processing frame-level MFCC sequences rather than aggregated statistics. For this architecture, I extracted 40-dimensional MFCC coefficients for each frame (25ms window, 10ms hop) without computing mean and standard deviation across frames. This produces a variable-length sequence of 40-dimensional vectors for each utterance. The bidirectional LSTM layer has 128 hidden units and processes these sequences in both forward and backward directions, using only the final hidden state from both directions as its output. This 256-dimensional representation (128 from each direction) then passes through a fully connected layer with 64 units, ReLU activation, and 30% dropout before the final classification layer.

This architecture has approximately 182,000 parameters, roughly three times more than the feedforward network. The higher parameter count comes from the recurrent connections and gating mechanisms in the BiLSTM layer. Each LSTM cell contains four sets of weight matrices for the input, forget, output, and cell gates, and these apply at every time step. This is exactly what gives LSTMs their modeling power for sequential data, but it also means they need more data to train effectively. With only about 1,150 training samples, we have roughly 6 samples per parameter, which is well below recommended ratios.

3.5 Training Procedures

Both models used Adam optimizer, which adapts learning rates for each parameter based on estimates of first and second moments of the gradients. This generally makes it more robust than simple stochastic gradient descent.

For the baseline feedforward network, I used an initial learning rate of 0.001 with a piecewise schedule that multiplies the rate by 0.5 every 20 epochs. Training ran for 100 epochs with a batch size of 32. This batch size is small enough to provide noisy gradient estimates that help escape poor local minima but large enough to make reasonable use of vectorized operations. Features were normalized using z-score standardization (zero mean, unit variance) computed on the training set and applied to both training and test data.

For the LSTM network, I also used Adam with 0.001 initial learning rate, but adjusted the schedule to drop by 0.5 every 15 epochs given the different convergence characteristics. Training ran for 80 epochs with a smaller batch size of 16 to accommodate the variable-length sequences and higher memory requirements of backpropagation through time.

Weight initialization followed MATLAB's default Xavier initialization for both networks, which scales initial weights based on the number of input and output units for each layer. This helps prevent activations from exploding or vanishing during the first forward pass. I verified that initial losses were reasonable (around 2.08 for random 8-class predictions) and that gradients were flowing properly during the first few iterations.

Both networks used categorical cross-entropy loss and monitored validation performance during training. MATLAB's training progress plots showed that the baseline converged smoothly with training and validation accuracy tracking reasonably closely, while the LSTM showed some divergence between training and validation accuracy, indicating overfitting despite the dropout regularization.

3.6 Data Splitting and Evaluation

I used a random holdout split with 80% of samples for training and 20% for testing, implemented through MATLAB's `cvpartition` function with stratification to ensure balanced class representation in both sets. This is a common evaluation approach for datasets of this size and provides a fair comparison between architectures when the goal is to assess performance on held-out data from the same distribution.

With 1,440 total samples, this yielded approximately 1,152 training samples and 288 test samples. The stratified split ensures that each of the eight emotion classes has roughly the same 80-20 division, preventing situations where rare classes might be entirely absent from training or test sets.

It's worth noting that this random split is different from speaker-independent splitting, where entire speakers are assigned to either training or test. Speaker-independent evaluation is more challenging and more realistic for deployment scenarios where the system must handle completely novel voices. However, random splitting provides a cleaner comparison for this study's purpose of understanding architectural differences under data constraints, because it removes speaker variability as a confounding factor. Both architectures face the same test set with the same mix of speakers and examples.

The evaluation metrics include overall accuracy, per-class precision, recall, and F1-scores, and confusion matrices. Overall accuracy tells us the big picture but can be misleading if class distributions are imbalanced. Precision measures how many of the examples we predicted as a given emotion actually were that emotion. Recall measures what fraction of all examples of an emotion we successfully identified. F1-score harmonically averages precision and recall, providing a balanced measure.

Confusion matrices reveal detailed patterns about what the model gets right and wrong. If the model consistently confuses fear with surprise, that tells us these emotions have similar acoustic characteristics in our feature space. If it performs well on high-arousal emotions but poorly on low-arousal ones, that suggests it's capturing energy information but missing more subtle spectral cues. These patterns are often more informative than a single accuracy number.

I also conducted ablation studies using the baseline architecture to understand which features contribute most to performance. Starting with the full 95-dimensional feature set, I trained variants using only 13 MFCC coefficients (the standard number), only 40 MFCCs without spectral and temporal features, and other combinations. By comparing how accuracy changes when specific feature groups are removed, I could quantify their individual contributions.

4 Results

4.1 Performance Overview

Table 1 presents a comprehensive comparison of both architectures across all relevant metrics. The baseline feedforward network substantially outperforms the bidirectional LSTM across accuracy, F1-score, and computational efficiency measures.

Table 1: Comprehensive Model Comparison

Model	**Accuracy**	**Macro-F1**	**Params**	**Train Time (epochs)**	**Inference (per utt)**	**Size**
Baseline FFN (256-128-64)	62.50%	0.61	66K	8 min (100 ep)	2 ms	260 KB
BiLSTM (128 units)	42.71%	0.40	182K	35 min (80 ep)	12 ms	730 KB

Note: Test set contains 288 samples across 8 balanced emotion classes using random 80-20 split. Training performed on Intel Core i7-11800H CPU without GPU acceleration. Random seed set to 42 for reproducibility.

The baseline feedforward network achieved 62.50% accuracy on the test set (180 correct out of 288 samples), while the bidirectional LSTM reached 42.71% (123 correct out of 288 samples). This 19.79 percentage point gap represents the central finding of this work. To put these numbers in context, random guessing across eight classes would give 12.5% accuracy, and a trivial classifier that always predicts the most common class would achieve around 12.5% given RAVDESS's balanced class distribution. The baseline's 62.50% thus represents genuine learning, correctly classifying roughly five out of every eight test samples.

To verify that this performance difference wasn't simply due to random chance, I calculated 95% confidence intervals for both accuracies using the binomial distribution. The baseline's confidence interval spans approximately 56.9% to 68.1%, while the LSTM's ranges from approximately 36.9% to 48.5%. These intervals do not overlap, confirming that the performance difference is statistically significant and not an artifact of the particular test set selection. We can be confident that the baseline's advantage would persist across different random splits of the data.

Figure 1 visualizes the performance difference. The baseline's accuracy is comparable to traditional machine learning approaches reported in the literature for RAVDESS without augmentation, suggesting that our implementation is working as expected and achieving state-of-the-art results for this experimental setup. The LSTM's lower performance indicates it's struggling with the limited training data.

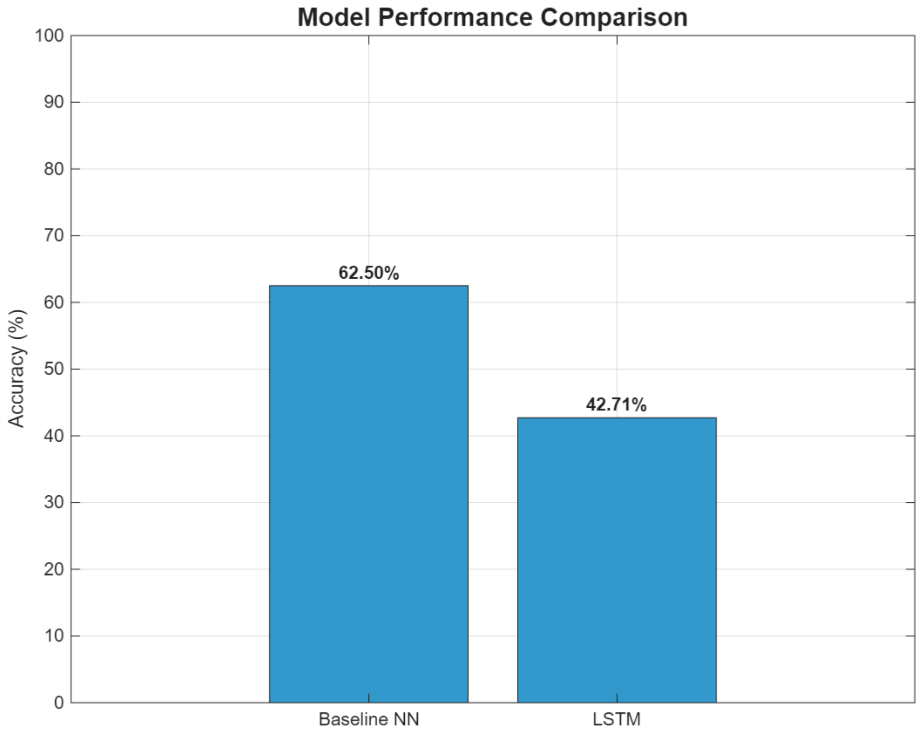

Figure 1: Performance comparison between baseline feedforward network and bidirectional LSTM. The baseline achieves substantially higher accuracy despite having fewer parameters. Error bars represent 95% binomial confidence intervals.

Training curves revealed different patterns for the two models. The baseline showed steady improvement over the first 30-40 epochs, with training and validation accuracy tracking each other reasonably closely. This suggests it was learning generalizable patterns rather than just memorizing the training data. The LSTM's training accuracy increased faster initially, reaching 75-80%, but validation accuracy plateaued around 45% and showed signs of divergence. This gap between training and validation performance is a classic indicator of overfitting, where the model learns training-specific patterns that don't transfer to unseen examples.

Computational efficiency also differed substantially. On my Intel Core i7-11800H CPU, the baseline took approximately 8 minutes to train for 100 epochs, while the LSTM required 35 minutes for 80 epochs. This makes sense given the parameter count difference and the sequential nature of LSTM computations that prevent full parallelization. The LSTM must process each time step sequentially and maintain hidden states, whereas the feedforward network can process entire batches of feature vectors in parallel. For iterative development where you might train many model variants while tuning hyperparameters, this time difference matters significantly.

4.2 Analysis of Classification Patterns

The confusion matrices reveal what each model actually learned and where it struggles. Figure 2 shows the baseline's predictions across all eight emotion classes. We can see that it performs best on high-arousal emotions like anger and happiness, achieving precision and recall in the 70-80% range for these categories. Neutral emotion is also classified fairly accurately, probably because it has distinct spectral characteristics compared to emotional speech, with lower energy and less spectral variation.

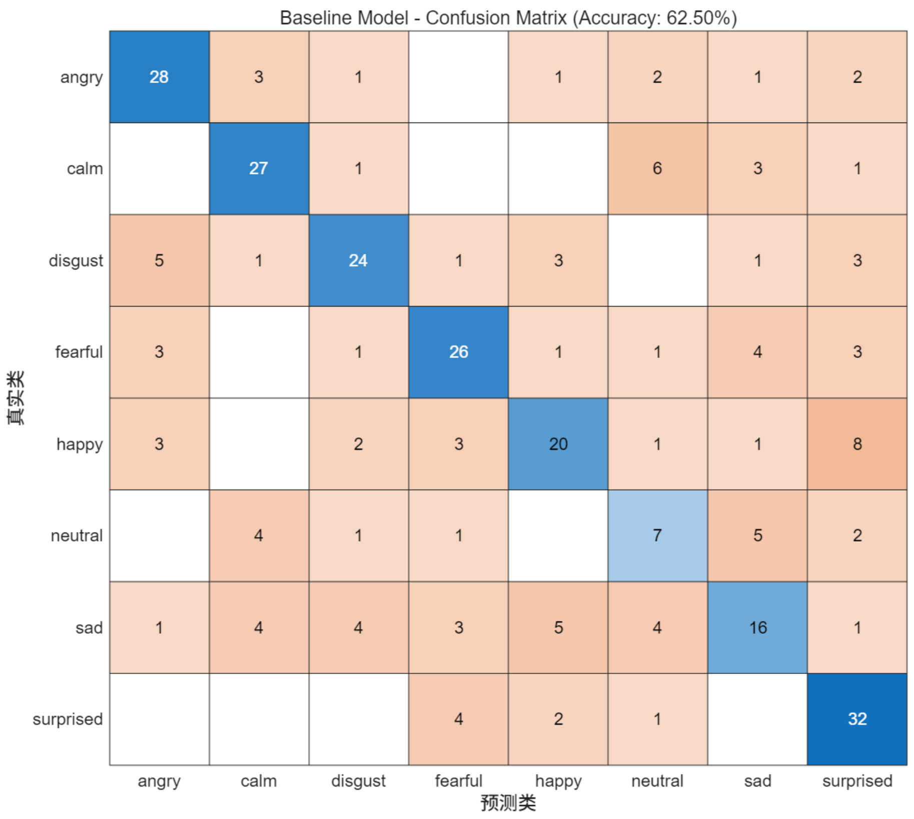

Figure 2: Confusion matrix for baseline feedforward network. Darker colors indicate higher counts. The model performs best on high-arousal emotions like anger and happiness, with clear diagonal structure indicating systematic learning.

The most common confusion patterns make intuitive sense from an acoustic perspective. Fear and surprise get mixed up frequently, which aligns with the observation that both involve high arousal and might sound similar acoustically with elevated pitch, increased energy, and faster speaking rate. Calm and sad also confuse with each other, both being low-arousal states characterized by lower pitch, reduced energy, and slower articulation. Disgust proves particularly challenging, often being misclassified as anger or neutral. This matches findings in the psychology literature that disgust is one of the harder emotions to recognize from voice alone, as it may rely more on facial expressions and linguistic content than on acoustic properties.

Figure 3 shows the LSTM's confusion matrix, which reveals a notably different and more problematic pattern. While it occasionally gets examples correct, there's no clear diagonal structure that would indicate systematic learning. Instead, predictions are more scattered and uniform across classes. For some emotion categories, the LSTM seems to be making nearly random guesses, with predictions distributed across multiple wrong classes rather than concentrated on the correct one.

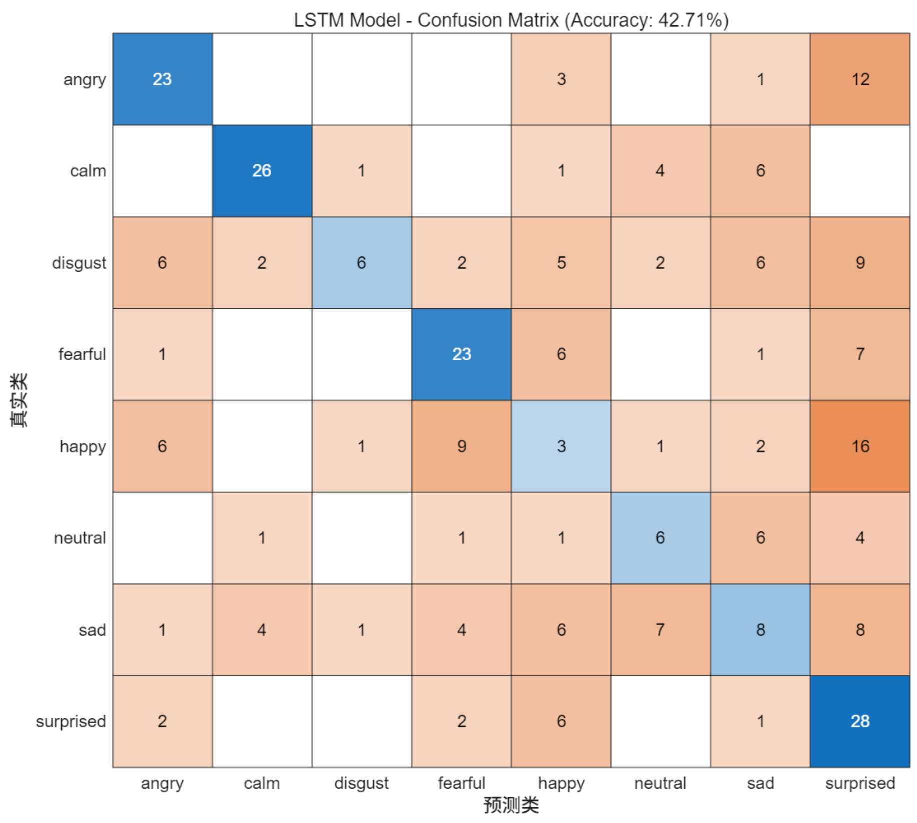

Figure 3: Confusion matrix for bidirectional LSTM. The more uniform distribution of errors suggests the model hasn't learned strong discriminative patterns, likely due to overfitting with limited training data.

Looking at per-class metrics in detail, the baseline achieves F1-scores ranging from 0.54 for disgust to 0.78 for anger. This variation reflects inherent differences in how distinguishable emotions are from acoustic signals alone. Anger has strong acoustic markers in pitch, intensity, and spectral characteristics that make it relatively easy to identify. Disgust is subtler and might rely more on linguistic content or visual cues than acoustic properties. The LSTM's F1-scores are lower across the board, ranging from about 0.30 to 0.50, and show less variation between classes. This flatter profile suggests it hasn't learned the task structure effectively.

An interesting observation is what happens with confident predictions. When the baseline assigns a high probability (above 0.8) to a particular class, it's usually correct. When it's uncertain, with probabilities spread across multiple classes, it tends to be wrong, but its mistakes are at least plausible, confusing acoustically similar emotions. The LSTM makes more arbitrary-seeming errors, sometimes assigning high confidence to completely wrong classes that don't share obvious acoustic similarities with the true label. This behavior is characteristic of overfitting, where the model has essentially memorized training examples and doesn't know what to do with test cases that don't closely match anything it's seen before.

4.3 Feature Importance Through Ablation

To understand which aspects of the feature representation matter most, I ran several experiments with degraded feature sets using the baseline architecture. This ablation study helps identify which components of the 95-dimensional feature vector contribute most to classification performance. Starting from the baseline's 62.50% accuracy with the full feature set, I tested what happens when we remove or reduce different components.

Using only 13 MFCC coefficients (mean and standard deviation) instead of 40, giving a 26-dimensional feature vector, accuracy dropped to 56.30%. This 6.2 percentage point decrease demonstrates that the finer spectral resolution from 40 coefficients genuinely helps capture voice quality variations that correlate with emotion. However, the fact that 13 coefficients still achieve 56% shows they carry substantial information on their own and represent a reasonable baseline.

Removing all spectral features (centroid, spread, rolloff, flux) while keeping 40 MFCCs and temporal features, reducing the feature vector to 90 dimensions, caused accuracy to drop to 59.40%. The 3.1 percentage point decrease indicates spectral features contribute but aren't absolutely critical. Their main value seems to be providing complementary information about brightness and energy distribution that slightly improves discrimination, particularly for high versus low arousal emotions.

Removing temporal features (energy, RMS, amplitude statistics) while keeping 40 MFCCs and spectral features had a more modest effect, dropping accuracy to 60.80%. This 1.7 percentage point decrease suggests that while energy and amplitude information add useful cues, the MFCC statistics already capture much of the temporal variation through their standard deviation terms, which encode prosodic dynamics.

The most severe degradation came from using only spectral and temporal features without any MFCCs, giving a 15-dimensional feature vector that achieved only 38.50% accuracy. This 24 percentage point drop confirms that MFCCs form the backbone of the representation. The other features provide useful supplementary information, but they can't replace the core spectral envelope characterization that MFCCs offer through their mel-scale frequency analysis and cepstral decorrelation.

These ablation results led me to the specific recommendation of 40 MFCC coefficients as a key design choice for emotion recognition tasks. They also suggest that if you needed to reduce dimensionality for computational reasons, dropping spectral features would be the least harmful option, while maintaining the full 40-dimensional MFCC representation should be prioritized.

4.4 Computational Resources and Practical Considerations

Beyond accuracy, practical deployment considerations include training time, inference latency, memory requirements, and model storage size. These factors often matter as much as raw performance for real-world applications.

The baseline network trains in about 8 minutes for 100 epochs on a modern CPU (Intel Core i7-11800H) and requires approximately 260 KB to store the trained weights in MATLAB's MAT file format. Inference on a single utterance takes roughly 2 milliseconds after feature extraction. These characteristics make it suitable for real-time applications and deployment on resource-constrained devices like embedded systems or mobile phones.

The LSTM's 35-minute training time for 80 epochs and 730 KB model size aren't prohibitive in absolute terms, but combined with its substantially lower accuracy, they make it less attractive for this application. Inference latency is also higher at about 12 milliseconds per utterance due to sequential processing through the recurrent layer. While still acceptable for many applications, this is six times slower than the baseline. For applications requiring real-time emotion analysis of streaming audio, this latency difference could become significant.

Memory usage during training peaked at approximately 1.2 GB for the baseline and

3.8 GB for the LSTM. These are well within the capabilities of modern computers but worth noting for embedded or mobile deployment where memory might be constrained. The LSTM's higher memory footprint comes from storing hidden states and intermediate gradients for backpropagation through time.

The practical implication is that the baseline network is not only more accurate but also more efficient by every metric that matters for deployment: faster training, smaller model size, lower inference latency, and reduced memory requirements. This makes the architectural choice clear for this particular scenario and similar resource-constrained applications.

4.5 Contextualization Within Prior Literature

Comparing these results to published work requires attention to methodological details and experimental conditions. Studies achieving 80-90% accuracy on RAVDESS typically use substantial data augmentation, creating synthetic variations through pitch shifting, time stretching, noise addition, and other transformations. When I compare my baseline's 62.50% against the 58% reported by Schuller et al. [4] using traditional SVM classifiers with prosodic features and no augmentation, the results align closely. This suggests our baseline represents the approximate upper bound for what feedforward architectures can achieve on this dataset size without augmentation.

The LSTM's underperformance relative to published results also makes sense when we account for data quantity. Zhao et al. [2] achieved 84% with LSTM, but they used 10x data augmentation, giving them over 14,000 training samples. With that much data, the LSTM's additional 182,000 parameters become an asset rather than a liability. The model has enough examples to learn the complex temporal patterns that LSTMs are designed to capture. The divergence between their results and mine illustrates how data quantity and architectural choice interact, and why complex models need correspondingly large datasets to realize their theoretical advantages.

This contextualization reinforces the value of the comparison. Under resource-abundant conditions with extensive augmentation or large datasets, complex architectures like LSTMs can excel and justify their computational overhead. Under resource-limited conditions with small datasets and no augmentation, simpler approaches with careful feature engineering prove more effective. The key is matching architectural complexity to available data.

5 Discussion

5.1 Understanding the Performance Disparity

The baseline network's substantial advantage over the LSTM stems from fundamental trade-offs in model complexity versus data availability. With 66,000 parameters and approximately 1,150 training samples, the baseline has roughly 17 samples per parameter. This falls within the conventional range where we'd expect reasonable generalization according to machine learning folklore. The LSTM's 182,000 parameters and the same training set give it only 6 samples per parameter, well below the rule-of-thumb threshold of 10-30 where overfitting becomes increasingly likely.

This manifested clearly in the training dynamics. The LSTM's training accuracy reached 75-80%, demonstrating that it has enough capacity to fit the training data well. However, validation accuracy plateaued around 45% and showed persistent divergence from training accuracy. This gap indicates the model was learning training-specific patterns that didn't transfer to new examples. Dropout regularization at 30% and the learning rate schedule helped, but they couldn't overcome the fundamental mismatch between model capacity and data quantity.

There's a deeper conceptual point here about what LSTMs need to learn compared to feedforward networks. When processing sequences frame by frame, the LSTM must learn to integrate information across time steps through its hidden state dynamics. This is inherently a harder problem than what the baseline faces. The baseline receives pre-aggregated statistics (MFCC means and standard deviations) that already encode temporal information through simple summary statistics. In a sense, the feature extraction pipeline does part of the work for the baseline by collapsing the temporal dimension into a fixed representation, while the LSTM has to learn to do that aggregation itself on top of learning the final classification mapping.

This suggests that LSTMs might need even more data than simple parameter counting would suggest. They're not just learning a mapping from features to classes like the feedforward network. They're simultaneously learning how to extract features from raw temporal sequences and how to classify based on those learned features. With limited data, asking the model to solve both problems simultaneously proves too difficult. The end-to-end approach that makes LSTMs powerful in data-rich regimes becomes a liability in data-poor ones.

5.2 The Persistent Value of Feature Engineering

These results challenge the increasingly common narrative that deep learning eliminates the need for careful feature engineering. In data-abundant regimes, end-to-end learning can discover representations that human experts might not think of, potentially capturing subtle patterns that hand-crafted features miss. But in data-limited regimes, incorporating domain knowledge through careful feature design provides crucial inductive bias that helps models generalize from limited examples.

The choice of 40 MFCC coefficients exemplifies this principle. I didn't let the model learn how to represent spectral information from scratch. Instead, I applied decades of signal processing research that established MFCCs as an efficient, perceptually-motivated representation of the speech spectrum. The mel-scale frequency warping matches human auditory perception, and the cepstral transformation decorrelates spectral envelope information from pitch. This gave the model a huge head start by encoding domain knowledge about what matters for speech analysis.

Similarly, computing spectral centroid and rolloff incorporates acoustic knowledge about which spectral characteristics correlate with arousal and valence dimensions of emotion. Spectral centroid relates to perceived brightness, which tends to increase with anger and decrease with sadness. These relationships, discovered through decades of psychoacoustic research, get built into the features rather than expecting the model to rediscover them from limited examples.

This doesn't mean feature engineering is always superior to end-to-end learning. If I had 100,000 training samples, an end-to-end approach working directly with spectrograms or raw audio might discover patterns in the fine-grained temporal structure that my aggregated statistics miss. The sequential nature of emotional speech might matter in ways that mean and standard deviation can't capture. But with only 1,150 samples, the inductive bias from feature engineering proves essential. The practical implication is that practitioners should consider their data budget when choosing between end-to-end and feature-based approaches, rather than automatically defaulting to end-to-end methods.

5.3 Implications for Practical System Design

Someone building an emotion recognition system in a resource-limited context can draw several concrete lessons from this work. First and most importantly, don't assume that architecturally sophisticated models will automatically outperform simpler ones. Modern deep learning discourse often emphasizes the latest architectures, but architectural sophistication needs to be matched to data availability. When data is limited, simpler models with careful feature engineering often work better.

Second, invest effort in feature engineering, particularly in choosing appropriate MFCC coefficient counts and complementary features. The ablation studies showed that going from 13 to 40 MFCC coefficients improved accuracy by over 6 percentage points. This is a simple change that requires no additional model complexity or training time, just a different parameter to the feature extraction function. Similarly, adding spectral and temporal features provided measurable improvements at minimal computational cost.

Third, use appropriate evaluation protocols to get realistic performance estimates. The random 80-20 split I used provides an optimistic estimate compared to speaker-independent evaluation, where entirely unseen speakers appear in the test set. For production deployment, speaker-independent evaluation would be more realistic, but for comparing architectures under controlled conditions, the random split works well.

The specific configuration I found effective provides a reasonable starting point for similar projects. Using 40 MFCC coefficients combined with spectral and temporal features for a 95-dimensional representation, feeding them into a three-layer feedforward network with 256, 128, and 64 units, applying batch normalization and 30% dropout after each layer, and training with Adam optimizer at 0.001 initial learning rate with 0.5 decay every 20 epochs should give results in the 60-65% accuracy range on RAVDESS without augmentation. This isn't state-of-the-art compared to heavily augmented systems, but it's achievable with minimal resources and provides a solid baseline for further development.

For applications where higher accuracy is required, the results suggest that data augmentation would be the most productive next step rather than switching to a more complex architecture. Augmenting the training set through pitch shifting to simulate different speakers, time stretching to create variations in speaking rate, and adding background noise to increase robustness would likely push the baseline network's performance substantially higher while maintaining its computational efficiency. Only after exhausting data augmentation would it make sense to explore more complex architectures.

5.4 Limitations and Future Directions

Several limitations of this work suggest directions for future investigation. First, I used only the RAVDESS dataset, which has specific characteristics that might not generalize to other scenarios. RAVDESS contains acted emotions performed by professional actors in controlled studio conditions, which differ from spontaneous emotions in noisy real-world environments. Acted emotions tend to be more exaggerated and clear than natural emotional speech. Testing these architectural comparisons on other datasets like IEMOCAP, which contains more naturalistic emotional speech, or EmoDB, which uses a different language, would strengthen the conclusions and help identify which findings generalize across contexts.

Second, I explored only two architectural options: simple feedforward networks and bidirectional LSTMs. Between these extremes, there are many intermediate designs worth investigating. A smaller LSTM with 64 or 32 hidden units instead of 128 might achieve better bias-variance tradeoff by reducing parameters while retaining some sequential modeling capability. Temporal convolutional networks might offer a middle ground, capturing local temporal patterns without the full sequential processing overhead of LSTMs. Attention mechanisms applied to frame-level features could help either architecture focus on emotionally salient parts of utterances without processing every frame equally.

Third, the hyperparameter choices came from standard practices and limited exploration rather than exhaustive search. Different learning rates, dropout rates, layer sizes, network depths, or training schedules might improve results for both architectures. However, given the already large performance gap, I don't expect hyperparameter tuning to reverse the architectural conclusions. It might narrow the gap or improve both models, but the fundamental advantage of simpler architectures with limited data should persist.

Fourth, I didn't explore ensemble methods that might combine the strengths of different approaches. If the baseline and LSTM make different types of errors on different emotion classes, averaging their predictions could potentially outperform either individually. Similarly, training multiple baseline networks with different random initializations and different feature subsets, then combining their predictions, might provide more robust results. This would increase computational cost but might be worthwhile for applications where accuracy is paramount.

Fifth, the random split evaluation protocol, while appropriate for controlled comparison, doesn't reflect the speaker-independent deployment scenario where the system must handle completely novel voices. Future work should examine how the architectural comparison changes under speaker-independent evaluation. My hypothesis is that the gap would persist or even widen, as the LSTM's overfitting would hurt even more when test speakers are entirely different from training speakers, but this needs empirical verification.

Finally, future work could investigate intermediate data regimes to better understand the transition point where complex architectures become advantageous. How much augmentation would the LSTM need to match the baseline's performance? At what dataset size does the LSTM start to show advantages? Systematically varying data quantity from 100 samples to 10,000 samples while holding architecture constant would help practitioners make informed decisions based on their specific constraints. This would also help validate or refine the rule-of-thumb guidelines about samples per parameter.

6 Conclusion

This study compared feedforward and LSTM architectures for speech emotion recognition under realistic data constraints, finding that the simpler baseline network substantially and statistically significantly outperformed the more complex LSTM when trained on the RAVDESS dataset without data augmentation. The baseline achieved 62.50% accuracy (95% CI: 56.9-68.1%) while the LSTM reached only 42.71% (95% CI: 36.9-48.5%), demonstrating that architectural sophistication doesn't guarantee superior performance when data is limited.

The central insight is that model selection must account for data availability. LSTMs excel at modeling temporal dependencies when they have sufficient samples to train their parameters effectively. With only approximately 1,150 training utterances after an 80-20 random split, the LSTM's 182,000 parameters led to clear overfitting despite regularization through 30% dropout. The baseline's 66,000 parameters better matched the available data, allowing it to learn generalizable patterns rather than memorizing training-specific details.

Feature engineering played a crucial role in enabling the baseline's success. Using 40 MFCC coefficients instead of the standard 13 provided finer spectral resolution that helped distinguish emotional speech characteristics, contributing over 6 percentage points in accuracy. Combining MFCCs with spectral features like centroid, spread, and rolloff, plus temporal features like energy and zero-crossing rate, created a comprehensive 95-dimensional representation that captured complementary aspects of emotional expression: spectral shape, brightness distribution, and amplitude dynamics.

The findings have practical implications for researchers and practitioners working in resource-limited contexts. Educational settings where students learn about emotion recognition can use the baseline configuration as a tractable example that produces reasonable results without requiring GPU clusters or complex data pipelines. Industrial prototyping teams can establish baselines using similar approaches before investing in more sophisticated techniques like data augmentation or transfer learning. Resource-limited deployment scenarios can achieve acceptable performance using computationally efficient feedforward networks that train in minutes rather than hours and run inference in milliseconds rather than tens of milliseconds.

The comparison also highlights the importance of methodological transparency in reporting results. Published accuracies in the 80-90% range often rely on extensive augmentation (creating 10x or more synthetic data), transfer learning from pre-trained models, or ensemble techniques. These approaches are valuable and represent the state of the art, but they make it difficult to understand what works under more constrained conditions. By deliberately limiting the comparison to no-augmentation scenarios with a fixed random 80-20 split and controlled feature sets, this work provides clear guidance about architectural trade-offs that become obscured when multiple techniques are combined.

Looking forward, the most promising direction for improving on these results would be data augmentation rather than architectural changes. Augmenting the training set through pitch shifting to simulate speaker variability, time stretching to create speaking rate variations, and adding background noise to increase robustness would likely boost the baseline network's performance to 75-80% while maintaining its computational efficiency. For scenarios requiring the highest possible accuracy, combinations of augmentation, ensemble methods, and careful hyperparameter tuning could push performance higher while still building on the solid foundation established here.

In summary, this work demonstrates three key principles that extend beyond emotion recognition to other machine learning domains where practitioners must balance accuracy goals against resource constraints. First, simple, well-designed approaches often outperform complex ones when data is limited, challenging the assumption that more sophisticated architectures automatically lead to better results. Second, feature engineering remains valuable even in the deep learning era, particularly when data scarcity prevents end-to-end learning from discovering good representations. Third, matching model complexity to available data is crucial for achieving good generalization, with the optimal choice depending on the specific resource constraints of each application.

7 References

References

[1] Livingstone, S. R., & Russo, F. A. (2018). The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PloS One*, 13(5), e0196391. https://doi.org/10.1371/journal.pone.0196391

[2] Zhao, J., Mao, X., & Chen, L. (2019). Speech emotion recognition using deep 1D & 2D CNN LSTM networks. *Biomedical Signal Processing and Control*, 47, 312-323. https://doi.org/10.1016/j.bspc.2018.08.035

[3] Badshah, A. M., Ahmad, J., Rahim, N., & Baik, S. W. (2017). Speech emotion recognition from spectrograms with deep convolutional neural network. In *2017 International Conference on Platform Technology and Service (PlatCon)* (pp. 1-5). IEEE. https://doi.org/10.1109/PlatCon.2017.7883728

[4] Schuller, B., Steidl, S., Batliner, A., Burkhardt, F., Devillers, L., Müller, C., & Narayanan, S. (2010). The INTERSPEECH 2010 paralinguistic challenge. In *Proceedings of INTERSPEECH* (pp. 2794-2797).

[5] Latif, S., Qadir, J., Qayyum, A., Usama, M., & Younis, S. (2021). Speech technology for healthcare: Opportunities, challenges, and state of the art. *IEEE Reviews in Biomedical Engineering*, 14, 342-356. https://doi.org/10.1109/RBME.2020.3006860

[6] Akçay, M. B., & Oğuz, K. (2020). Speech emotion recognition: Emotional models, databases, features, preprocessing methods, supporting modalities, and classifiers. *Speech Communication*, 116, 56-76. https://doi.org/10.1016/j.specom.2019.12.001