CS 4375 Machine Learning Final Project Report

Comparative Analysis: Decision Trees vs Neural Networks for Facial Emotion Recognition

Team Members: Parthav Elangovan, Vasudev Nair

Task and Background

Problem Statement

Understanding human emotions in real-time has significant potential across multiple domains including education, mental health monitoring, and customer service. However, human emotions can be difficult to interpret accurately and consistently. This project aims to build and compare two automated systems that recognize facial expressions in real-time and classify them as neutral, happy, sad, and angry.

Objective

Compare Decision Trees versus Convolutional Neural Networks (CNNs) to determine which model performs better overall and under what conditions for facial emotion recognition tasks.

Dataset

FER-2013 Overview

FER = Facial Emotion Recognition. The dataset contains grayscale 48x48 pixel facial images labeled with seven emotion categories: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise.

Training and Test Split

For Decision Trees (4-class subset: Angry, Happy, Neutral, Sad):

Training Set Distribution:

- Angry: 3,995 images
- Happy: 7,215 images
- Neutral: 4,965 images
- Sad: 4,830 images
- Total: 21,005 images

Test Set Distribution:

- Angry: 958 images

- Happy: 1,774 images
- Neutral: 1,233 images
- Sad: 1,247 images
- Total: 5,212 images

For CNN:

- Trained on both the 4-class subset and full 7-emotion dataset

Dataset Challenges

- Low resolution (48x48 pixels) limits clarity and feature extraction
- Presence of watermarks on most images
- Class imbalance (Happy has nearly 2x more samples than Angry)

Approach

Decision Trees Methodology

Preprocessing

Raw images suffer from inconsistent lighting and low contrast. Our preprocessing method addressed these issues:

- Histogram Equalization: Redistributes pixel intensities across full [0-255] range
- CLAHE (Contrast Limited Adaptive Histogram Equalization): Enhances local features
- Edge Enhancement: Emphasizes facial feature boundaries

Augmentation

To prevent overfitting, each training image was augmented 4x:

- Horizontal Flip: Mirror image (faces work from any side)
- Rotation ±5°: Simulates slight head tilt variations
- Brightness ±20: Handles different lighting conditions

Result: 21,005 images became 84,020 augmented training samples

Feature Extraction (993 features)

Decision trees cannot process raw pixels directly. We engineered features across 6 categories:

1. **Brightness Features** - Multi-scale patch statistics (192 features)
- 6x6, 12x12, and 24x24 patches
- Extract mean, standard deviation, and range per patch
2. **Neighbor Comparison Features** - Local Binary Patterns (32 features)
- Compare each pixel with 8 neighbors

- Create 8-bit binary code per pixel
3. **Facial Geometry Features** - Regional statistics (14 features)
- Eye region: mean, std, median, min, max, percentiles
- Mouth region: same 7 statistics
- Eyes widen in surprise/fear, narrow in anger
- Mouth curves upward (happy) or downward (sad)
4. **Facial Shape Features** - HOG (689 features)
- Histogram of Oriented Gradients
- Divide image into 8x8 cells
- Captures facial feature shapes and structure
5. **Colors/Shading Features** - Histograms (48 features)
- 16-bin and 32-bin intensity histograms
- Angry faces tend to be darker overall
- Happy faces tend to be brighter
6. **Texture Features** - Gradient-based (18 features)
- Sobel and Scharr operators
- Detect horizontal edges (eyebrows, mouth)
- Detect vertical edges (nose, face outline)

Example feature values:

- mouth_curve = 0.85 (high upward curve indicates happy)
- texture_variance = 0.67 (high variance indicates angry)

Feature Selection

We then applied F-test to select the top 400 features from 993 total, eliminating redundant and noisy features while also retaining discriminative power.

Model Architecture

We trained two distinct decision tree-based ensemble models. The Random Forest model consisted of 500 individual trees with a maximum depth of 20 levels. This model uses a bagging approach where each tree is trained on a random bootstrap sample of the data and considers a random subset of features at each split. The training process is relatively quick, taking only minutes to complete. Predictions are made by having each tree vote independently, with the final prediction determined by majority vote across all 500 trees.

The Gradient Boosting model employed 300 trees with a more constrained maximum depth of 6 levels and a learning rate of 0.05. Unlike Random Forest's parallel training approach, Gradient Boosting builds trees sequentially, where each new tree focuses on correcting the errors made by the previous trees. This sequential error correction mechanism makes training significantly longer, requiring several hours to complete. However, this careful learning process with a low learning rate helps us prevent overfitting and gives better generalization performance.

Neural Networks Methodology

Preprocessing

Our raw image data contained noise and inconsistencies which posed a challenge to our neural network's convergence. We employed the following techniques to tackle this:

- Grayscale Conversion: Since color brings unnecessary details for emotion detection, we reduced the input depth to a single channel.
- Normalization: The pixel intensities were rescaled from [0-255] to [0-1]
- Resizing: The inputs were also resized to 96x96 dimensions.

Augmentation

To prevent overfitting, we implemented data augmentation through the following:

- Geometric Transformations: Rotation (25°), width/height shifts, shear, and zoom operations
- Horizontal Flip

Result: Our initial dataset had more examples with unique features

Feature Learning

The approach with the CNN learns discriminative features from raw pixel data.

1. **Low-Level Features - Block 1 (32 Filters)**

   Conv2D (3x3) -> ReLU -> BatchNormalization -> MaxPool

   This block operates similar to the edge detection filters in the Decision Tree approach by identifying primary visual cues such as vertical lines or horizontal curves.

2. **Mid-Level Features - Block 2 (64 Filters)**

   Two stacked Conv2D layers -> ReLU -> BatchNormalization -> MaxPool

   Because it combines low-level edges, this block learns to detect parts of the face, such as the shape of an eye or the corner of the mouth.

3. **High-Level Features - Block 3 (128 Filters)**

   Two stacked Conv2D layers -> ReLU -> BatchNormalization -> MaxPool

   This block aggregates shapes into abstract emotional representations, allowing it to distinguish between complex states like "surprise" (wide eyes + open mouth) versus "fear" (wide eyes + tense mouth.)

Model Architecture and Motivation

We decided to test out a deep CNN due to the fact that our data was visual and that facial expressions are spatial. CNN's can preserve spatial relationships through convolution and pooling operations.

System Components

- **Feature Extractor:** The three convolutional blocks described above act as the "eyes" of the system by compressing the 96x96 image into a dense representation.
- **Classifier (Dense Layers):** The top of the network has a Flatten layer which leads into the 1024-neuron Dense layer. This layer interprets the high-level features.
- **Output:** The Softmax layer produces a probability distribution for the classes.

Results and Discussion

Overall Performance

Decision Trees:

The Random Forest model achieved a test accuracy of 63.43%, with a consistent performance across the five-fold cross-validation showing an average of 63.44%. Examining the per-class performance reveals interesting patterns: the Angry class achieved precision of 0.58 and recall of 0.55, while the Sad class showed precision of 0.57 and recall of 0.52. The Neutral class performed slightly better with precision of 0.61 and recall of 0.66. However, the Happy class demonstrated the best performance across all emotions with precision of 0.71 and recall of 0.83, reflecting the distinctive nature of smiling facial expressions.

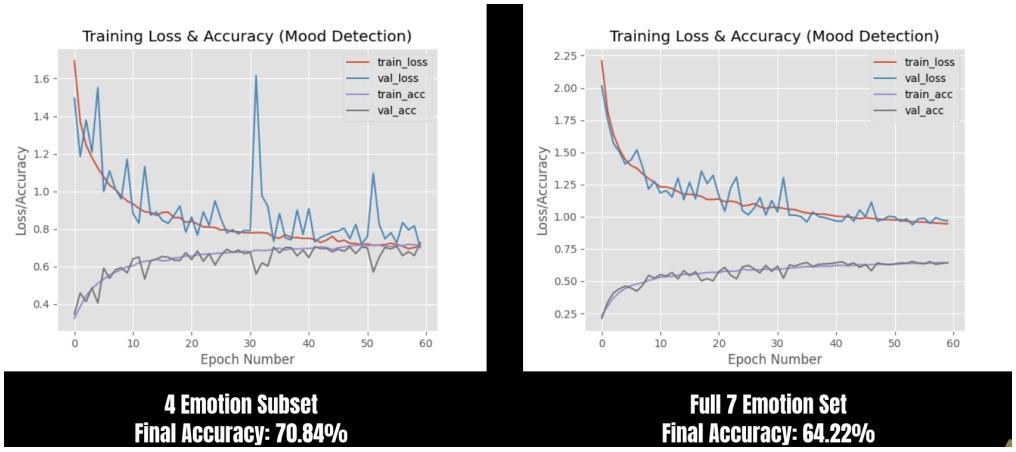
The Gradient Boosting model improved upon Random Forest with a test accuracy of 64.95%. Its per-class performance showed some variations from Random Forest: the Angry class had precision of 0.53 and recall of 0.49, while Sad achieved precision of 0.59 and recall of 0.58. The Neutral class showed precision of 0.63 and recall of 0.70. Most notably, the Happy class achieved even better performance than in Random Forest with precision of 0.79 and recall of 0.88, demonstrating Gradient Boosting's superior ability to capture the distinctive features of positive emotions because of each tree correcting the previous.

```
Random forest: 63.43%

Angry     P: 0.58 R: 0.55
Sad       P: 0.57 R: 0.52
Neutral   P: 0.61 R: 0.66
Happy     P: 0.71 R: 0.83

Gradient boosting - 64.95%

Angry     P: 0.53 R: 0.49
Sad       P: 0.59 R: 0.58
Neutral   P: 0.63 R: 0.70
Happy     P: 0.79 R: 0.88
```

```
1/4: Training Random Forest
  5-fold CV
    Fold 1: 63.17%
    Fold 2: 63.57%
    Fold 3: 63.23%
    Fold 4: 63.61%
    Fold 5: 63.62%
  CV: 63.44%
```

Model	Test Accuracy
Random forest	63.43%
Gradient boosting	64.95%

Neural Networks:



The **7-Class** CNN achieved a validation accuracy of **64.22%**, which shows a steady learning curve. We can see a stable convergence where the training and validation accuracy moved closely together; this suggests the dropout layers prevented overfitting as intended. This performance reveals the difficulty of distinguishing between subtle emotions like “Fear” vs “Surprise,” which have similar visual features such as widened eyes.

The **4-Class** Convolutional Neural Network improved significantly upon the 7-class results, reaching a validation accuracy of **70.84%**. By reducing the emotions to just the four most distinct emotions (Happy, Sad, Angry, Neutral), the model was able to form sharper decision boundaries. These results confirm that the CNN is utilized more efficiently when focusing on distinct emotions as opposed to specific variations.

Comparison to Literature

Authors of the FER-2013 dataset implemented a real-time CNN emotion recognition system using TensorFlow and Keras. With the full 7-emotion dataset, the IEEE study reports an overall accuracy of 65.97%. Our CNN achieved 64.22%, validating our implementation.

Key Findings

Analysis of model performance reveals several key findings. The Happy emotion consistently achieved the best performance across all models, with precision ranging from 0.71 to 0.79 and recall from 0.83 to 0.88. This superior performance stems from the distinctive visual features associated with smiling, particularly the upward curvature of the mouth and the activation of muscles around the eyes. In contrast, models struggled to separate Sad and Angry emotions, as both involve furrowed eyebrows and downturned mouth shapes that create similar visual patterns. Fear was often confused with Sad due to similar facial muscle tension patterns.

Lessons Learned

Technical Insights

Honestly, the biggest takeaway was realizing how much effort goes into manual feature extraction compared to just letting a CNN learn features automatically. We spent hours implementing and testing 993 different features using OpenCV's computer vision modules and still only hit 64.95% accuracy while the CNN easily reached 70.84%. It emphasizes on why deep learning has taken over computer vision.

The class imbalance was more frustrating than we expected. Happy had nearly twice as many training examples as Angry, and you could see it in the results, our models got really good at detecting happy faces (F1 of 0.88) but struggled with everything else. We tried class weighting to fix it, but that only helped a little. The real problem is just not having enough examples of some emotions, and there's only so much you can do about that without collecting more data.

Data augmentation was essential for preventing overfitting. Our 4-fold augmentation strategy using horizontal flips, small rotations, and brightness adjustments improved accuracy by approximately 5% and reduced the generalization gap substantially. Interestingly, CNNs benefited even more from augmentation, likely due to their ability to learn transformation invariances through convolutional architecture.

Individual Contribution

Parthav Elangovan (Decision Trees)

- Implemented complete preprocessing pipeline (histogram equalization, CLAHE, denoising, edge enhancement)
- Developed 4x data augmentation strategy (flip, rotation, brightness)

- Engineered all 993 features across 6 categories with OpenCV modules: brightness, neighbor comparison, facial geometry, facial shape, colors/shading, and texture
- Implemented F-test feature selection to reduce from 993 to 400 features
- Trained and optimized Random Forest (500 trees, depth 20) and Gradient Boosting (300 trees, depth 6, learning rate 0.05)
- Conducted 5-fold cross-validation and generated the performance metrics for the decision trees

Vasudev Nair (Neural Networks)

- Implemented the CNN preprocessing pipeline
- Created dynamic data augmentation strategy.
- Designed the 3-layer CNN architecture.
- Trained and optimized model configurations for both the 4-class and 7-class CNN's.
- Generated training/validation loss curves and analyzed performance.

Demo

A real-time CNN demo was implemented using webcam feed to detect emotions.

Link:
https://drive.google.com/file/d/1BOZ0FJ5c4eP0S50LahBIYtdU23wrHwFu/view?usp=sharing

Conclusion

This project demonstrates both the capabilities and limitations of decision tree ensembles for facial emotion recognition when compared against convolutional neural networks. Our Gradient Boosting model achieved 64.95% accuracy on the 4-class FER-2013 subset using 993 OpenCV-extracted features, while the CNN reached 70.84% through automatic feature learning. This 6% gap represents the current trade-off between interpretability and performance in emotion recognition systems.

Key Recommendation:

- For production systems prioritizing accuracy: Use CNNs
- For educational applications or explainable AI requirements: Use decision trees despite lower accuracy
- For edge deployment without GPU: Decision trees offer practical advantages

Both approaches achieved performance comparable to published baselines, validating our implementations and demonstrating that sophisticated feature engineering can enable classical machine learning methods to remain competitive on complex computer vision tasks. The real question is not which approach is objectively better, but which better serves better for specific requirements for accuracy, interpretability, and deployment constraints.