Facial Emotion Detection - Leveraging Deep Learning for Emotion Recognition

Presented by: Mohit Pammu

Date: April 2025

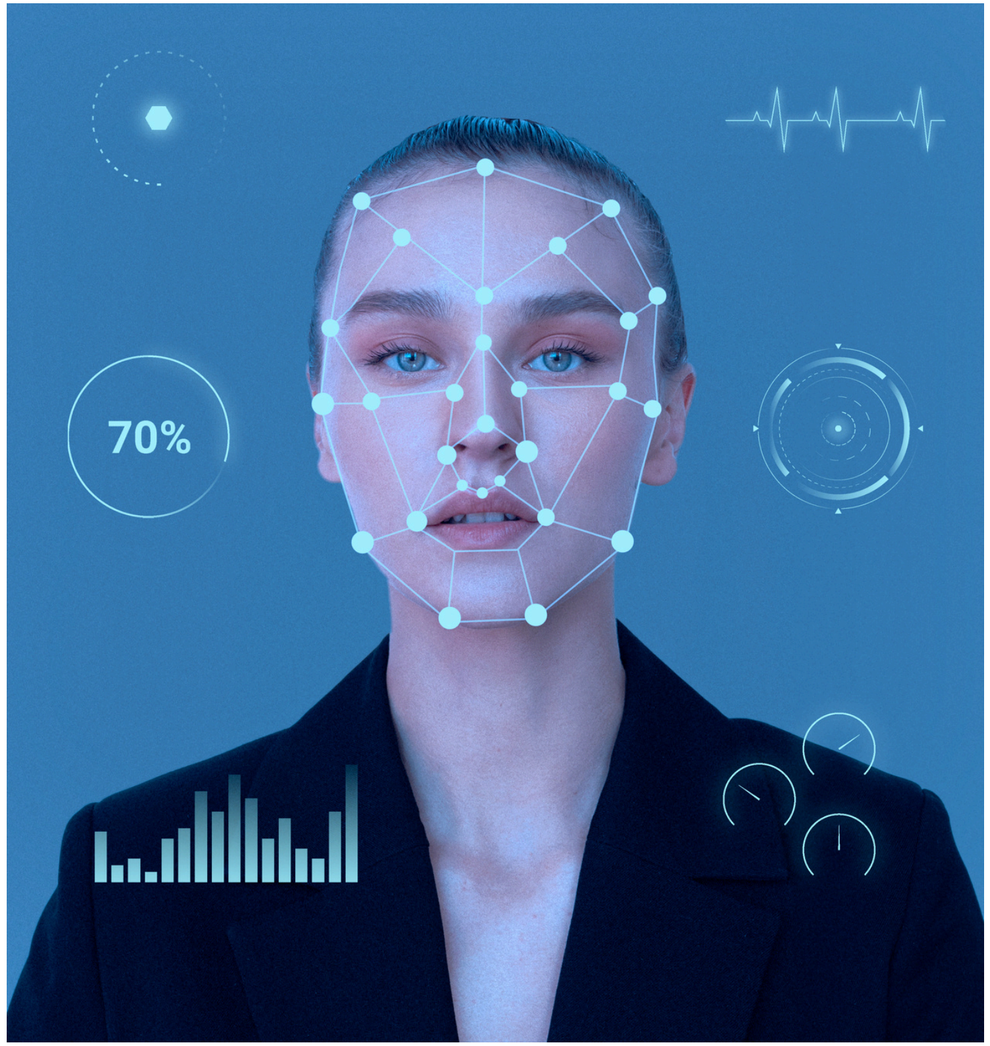
PROBLEM SUMMARY

CONTEXT

- Over 55% of human emotional communication is conveyed through facial expressions
- Affective Computing market projected to grow from $62.53B in 2023 to $388.28B by 2030

CHALLENGES

- Cultural and contextual factors influence emotion interpretation
- Ethical concerns around privacy and surveillance



OBJECTIVE

- Develop a robust deep learning model capable of performing multi-class emotion classification with high accuracy

APPLICATIONS

- Mental health monitoring (telehealth)
- Customer Service
- Human-Computer interaction
- Security and Surveillance
- Entertainment and Gaming

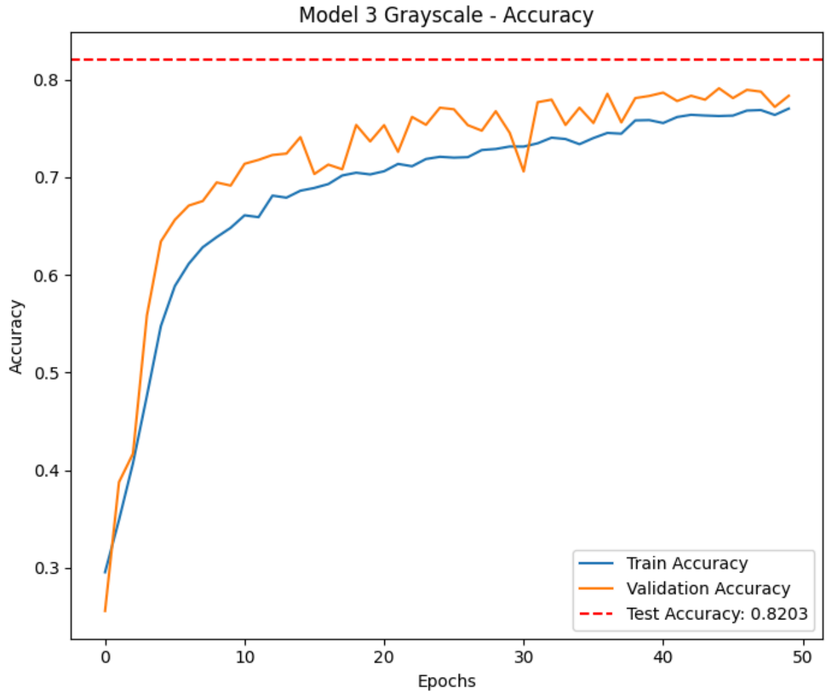
SOLUTION DESIGN

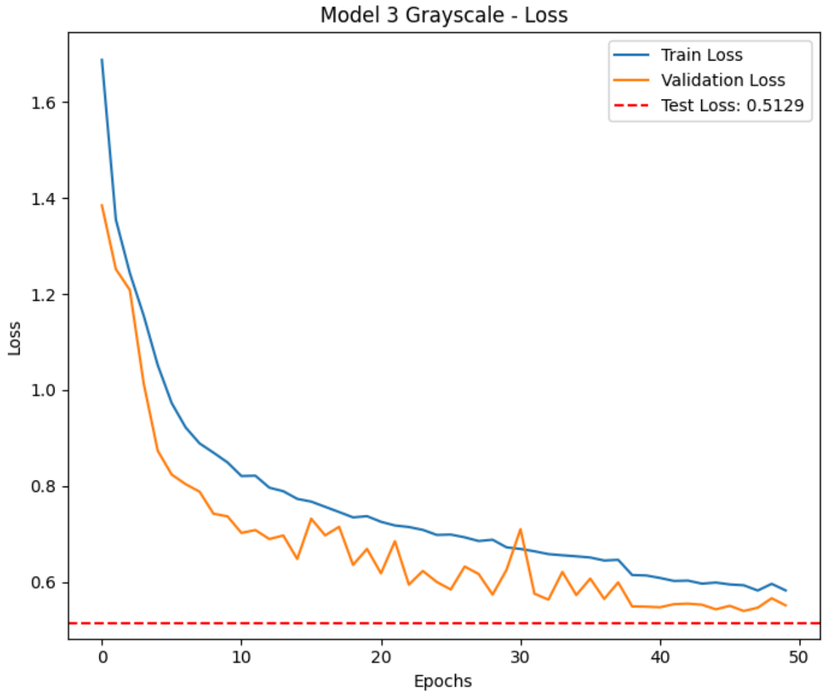
MODEL 3 ARCHITECTURE

- 3 convolutional blocks (32 → 64 → 128 filters)
- Dual convolutions per block
- Batch normalization + dropout (25—50%)
- Two dense layers (256, 128 neurons)
- Softmax activation (4-class)
- Loss: Categorical Crossentropy
- Optimizer: Adam (LR = 0.001)

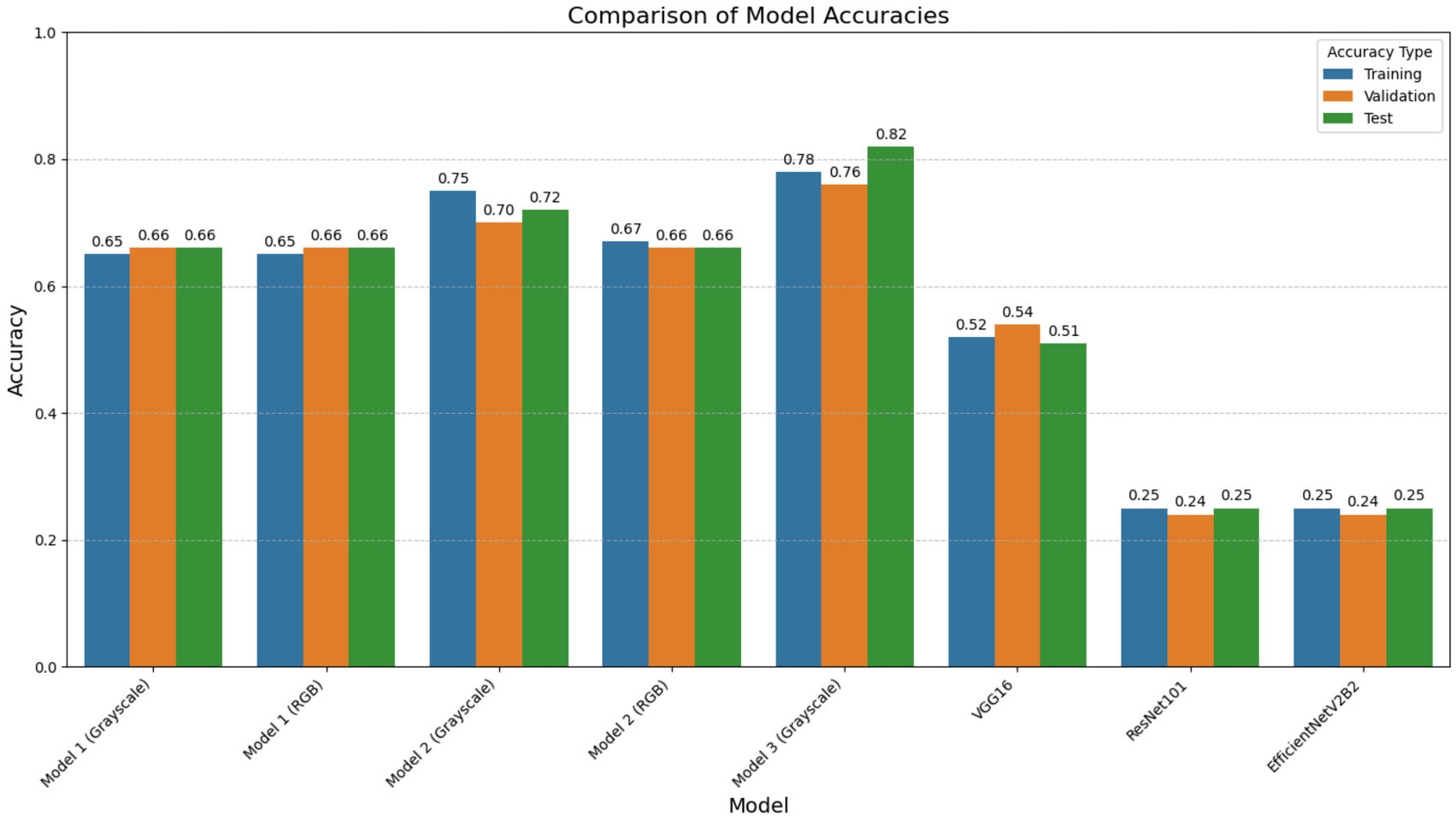
PERFORMANCE

- **Highest test accuracy (~82%)**
- **Efficient training time (~44 seconds per epoch)**
- **Superior to RGB-based transfer learning models (e.g., VGG16, ResNet, EfficientNet)**





MODEL COMPARISON GRAPH



ANALYSIS & KEY INSIGHTS

FINDINGS

- **Grayscale-specific models outperform RGB-based transfer learning**
- **Data augmentation improves robustness, especially for underrepresented classes**
- **Misclassification persists for overlapping emotions like 'sad' and 'neutral'**

IMPLICATIONS

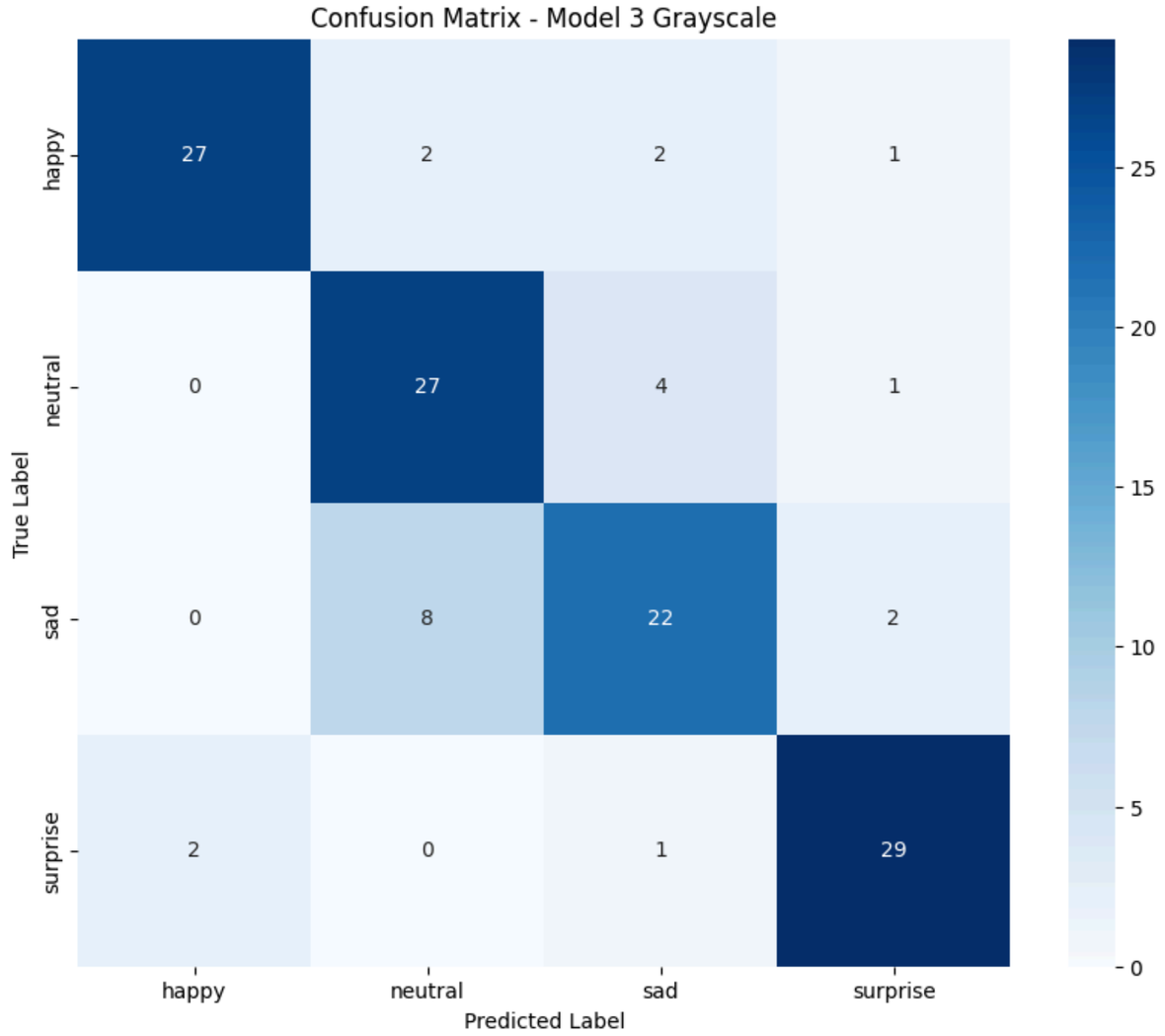
- **Static images are limited; temporal modeling or multi-modal inputs are needed**

TECHNICAL LIMITATIONS

- Class imbalance, especially for ‘surprise’ emotion
- Difficulty distinguishing visually similar emotions
- Transfer learning models struggled with grayscale inputs

REGULATORY COMPLIANCE

- GDPR (EU) and CCPA/CPRA (California)
- Healthcare - HIAA
- Education - FERPA
- Financial Services - documentation, audits



RECOMMENDATIONS FOR IMPLEMENTATION

ADOPTION OF MODEL 3

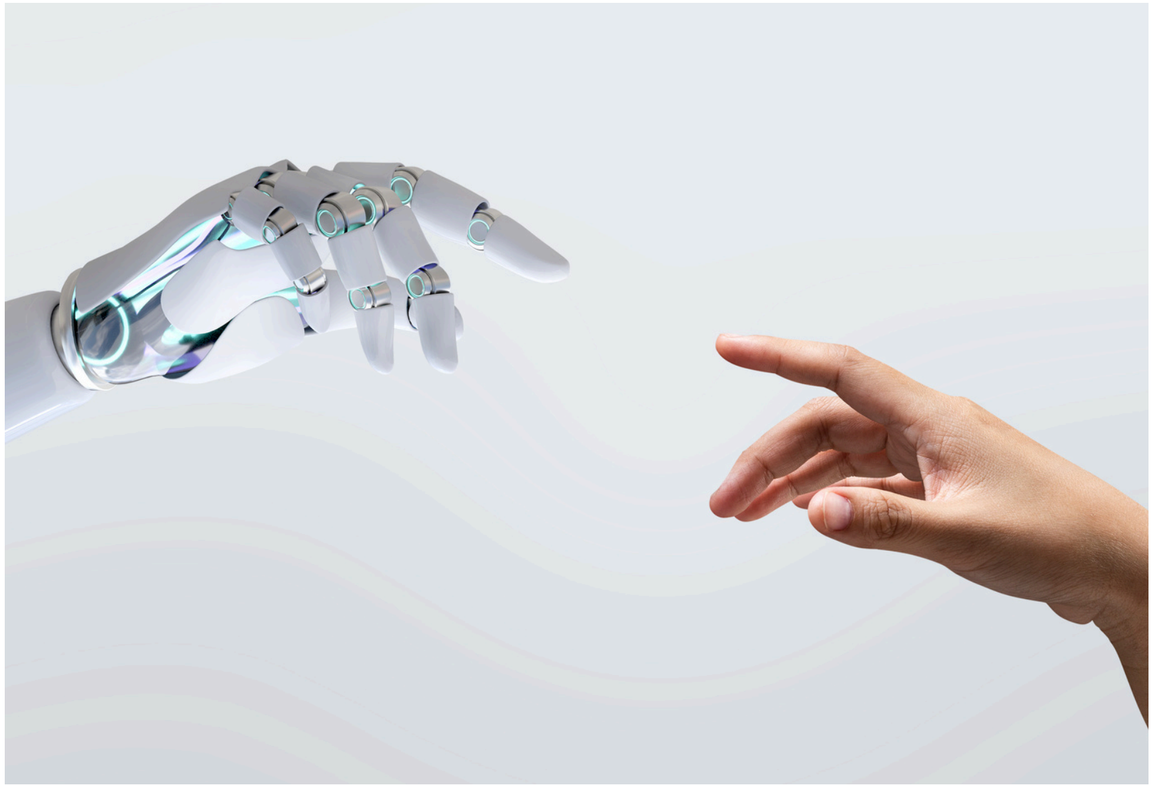
- Deploy as the production model for its accuracy and efficiency
- Use techniques like targeted data augmentation and weighted loss functions

INFRASTRUCTURE NEEDS

- Invest in scalable cloud or edge platforms
- Expand datasets to improve demographic diversity and reduce bias

ETHICAL CONSIDERATIONS

- Conduct bias audits and ensure transparency in deployment
- Collaborate with domain experts for responsible use



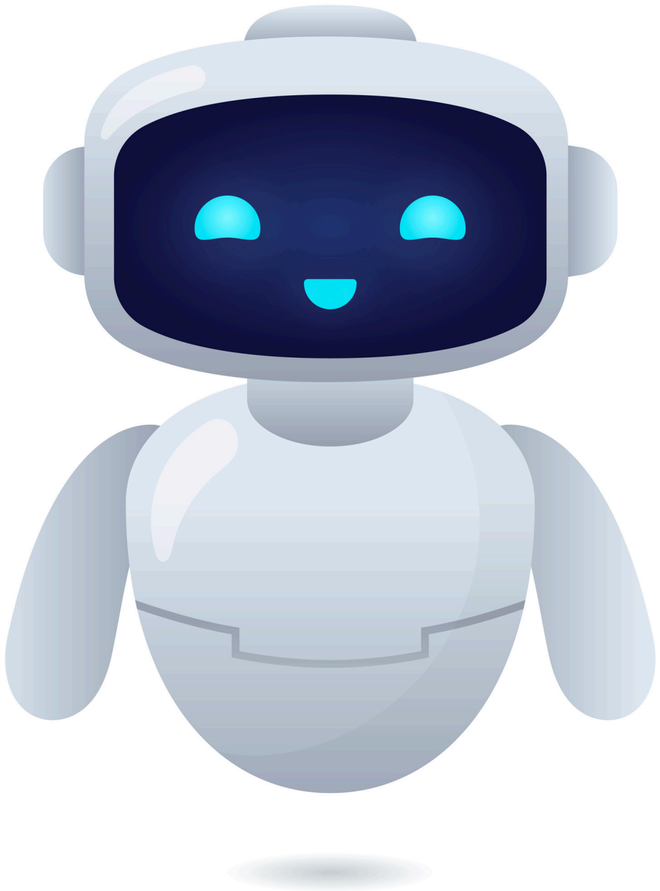
FUTURE DIRECTIONS

ENHANCEMENTS

- Explore multi-modal data integration (e.g., audio, video)
- Implement temporal modeling for dynamic emotion recognition
- Build APIs for modular deployment

TESTING AND VALIDATION

- Conduct real-world testing to refine and enhance performance
- Monitor and retrain with new data to ensure long-term accuracy



THANK YOU

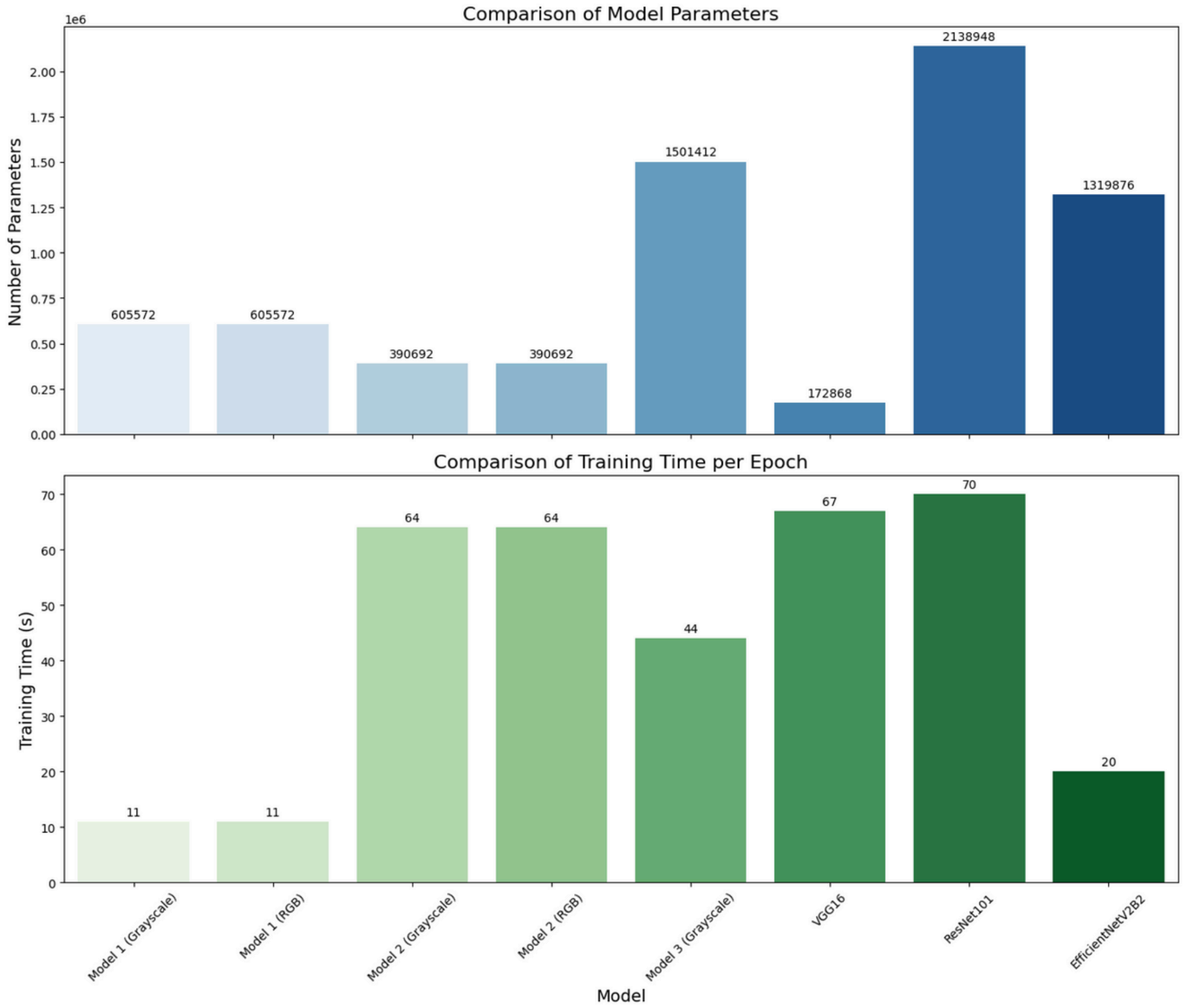
APPENDIX

MODEL COMPARISON CHART

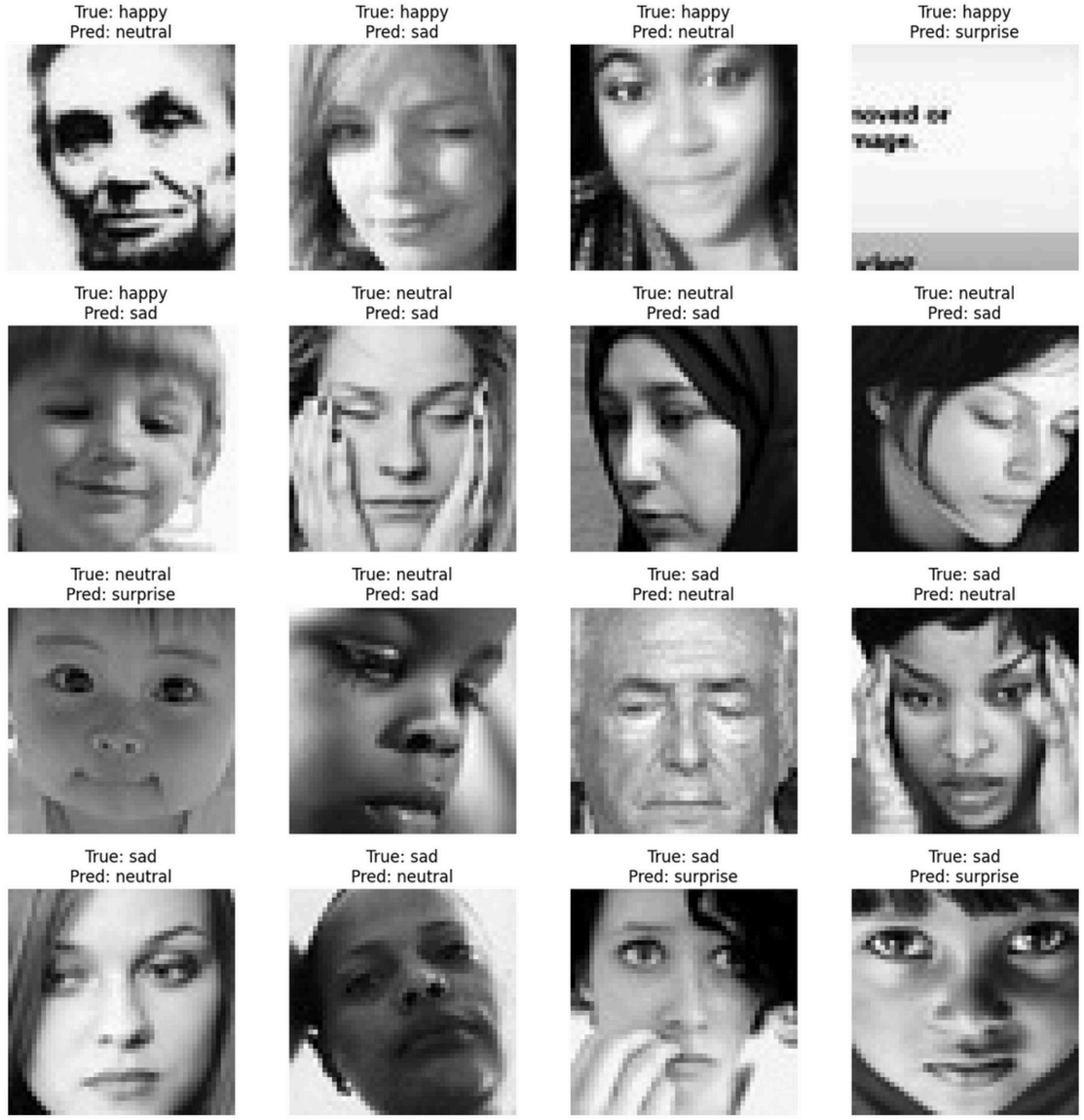
Model	Architecture Type	Input Mode	Test Accuracy	Training Time per Epoch	Notes
Model 1 (GS)	Simple CNN	Grayscale	66%	~11 sec	Basic architecture; limited feature extraction
Model 1 (RGB)	Simple CNN	RGB	66%	~11 sec	RGB input added no performance benefit
Model 2 (GS)	Medium CNN	Grayscale	72%	~64 sec	Better accuracy, slight overfitting observed
Model 2 (RGB)	Medium CNN	RGB	66%	~64 sec	RGB version underperformed vs grayscale
Model 3	**Complex CNN**	**Grayscale**	**82%**	**~44 sec**	**Best performance, good generalization, efficient and effective**
VGG16	Transfer Learning	RGB	51%	~67 sec	Poor performance on grayscale task; large model
ResNet101	Transfer Learning	RGB	25%	High	Computationally expensive; underfitting on grayscale
EfficientNetV2B2	Transfer Learning	RGB	25%	~20 sec	Fast but highly inaccurate; poor feature alignment with task

***GS = Grayscale**

MODEL COMPARISON: PARAMETERS AND TRAINING TIME



MODEL 3 MISCLASSIFICATIONS



MODEL 3 ARCHITECTURE

```python
# Define the grayscale CNN model 3
def model_3():
    model = Sequential()

    # First convolutional block - 32 filters
    model.add(Conv2D(32, (3, 3), padding='same', input_shape=(img_size, img_size, 1), activation='relu'))
    model.add(BatchNormalization())
    model.add(Conv2D(32, (3, 3), padding='same', activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    # Second convolutional block - 64 filters
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(BatchNormalization())
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    # Third convolutional block - 128 filters
    model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
    model.add(BatchNormalization())
    model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    # Fully connected layers
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(4, activation='softmax'))

    return model
```