Ain Shams University

Faculty of Engineering

Mechatronics Department

Machine Vision (CSE480s)

Milestone 1: Dataset Preparation and Model Development

Action recognition: UCF-101& Emotion recognition: FER-2013

Name	ID	SEC
Abdullah Mahmoud Abdel-Kareem	2100658	Sec 3
Musa Gamal	2100614	Sec 3
Abdalkarem wael abdalkarem	2101733	Sec 3

1- Emotion recognition

1. Problem Definition and Importance

➢ Problem Definition

The goal of this project is to develop a deep learning-based system capable of recognizing human facial emotions in real time. Facial emotion recognition is a critical task in human-computer interaction, as it allows systems to understand human affective states and respond appropriately. Emotions such as happiness, sadness, anger, fear, disgust, surprise, and neutrality can provide valuable cues in various applications, including:

Human-robot interaction

Mental health monitoring

Driver attention and safety systems

Social robotics and virtual assistants

➢ Importance

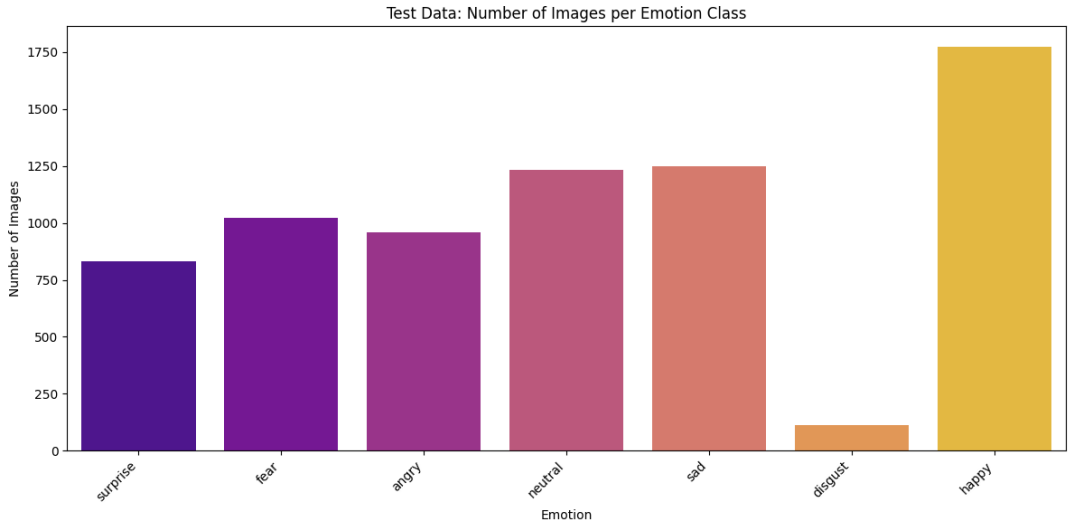
Automating emotion detection can significantly enhance user experience, assistive technologies, and security monitoring. Using CNNs for spatial feature extraction and LSTMs for temporal analysis ensures that both facial features and motion cues are captured efficiently. A robust model will enable real-time emotion detection, providing instant feedback for intelligent systems.

2. Data Cleaning and Preprocessing Steps

➢ Dataset

- Dataset: FER-2013
- Classes: 7 emotions [Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral]
- Training set: 28,709 images
- Testing set: 3,589 images

Data visualized:



- ➢ The train and test data looks not distributed normally it made a problem when train model as the model bias for large number of data, so that it should normalize and made augmentation for train data.
- ➢ The dataset with low resolution images made training and model accuracy not good.

➢ Data Cleaning

1. Verified that all images exist in the specified directories.
2. Counted images per class to detect class imbalance.
3. Identified missing or corrupted files (none in this dataset).

➢ Data Preprocessing

1. Resizing**:** Images resized to 48×48 pixels to reduce computation and maintain model efficiency.
2. Normalization: Pixel values scaled to [0,1] using rescale=1. /255.
3. Augmentation**:** Applied transformations to increase dataset diversity and reduce overfitting:
    a. Random rotations
    b. Width/height shifts
    c. Zoom
    d. Shear
    e. Horizontal flips
4. Class Weights: Computed to address class imbalance during training.

3. Methods and Algorithms

➢ Model Architecture

We built a deep Convolutional Neural Network (CNN) using Keras Sequential API:

- Block 1: Two Conv2D layers (64 filters, 3×3), BatchNorm, MaxPooling, Dropout
- Block 2: Two Conv2D layers (128 filters, 3×3), BatchNorm, MaxPooling, Dropout
- Block 3: Two Conv2D layers (256 filters, 3×3), BatchNorm, MaxPooling, Dropout
- Block 4: Two Conv2D layers (512 filters, 3×3), BatchNorm, MaxPooling, Dropout
- Dense Layers: Flatten → Dense (1024) → Dense (512) → Output (7 classes, softmax)

➢ The model is a multi-stage CNN consisting of four convolutional blocks (64 to 512 filters), each integrating Conv2D, BatchNorm, pooling, and dropout to enhance feature extraction and prevent overfitting. These features flow into dense layers of 1024 and 512 neurons, culminating in a softmax layer that predicts one of seven emotion classes.

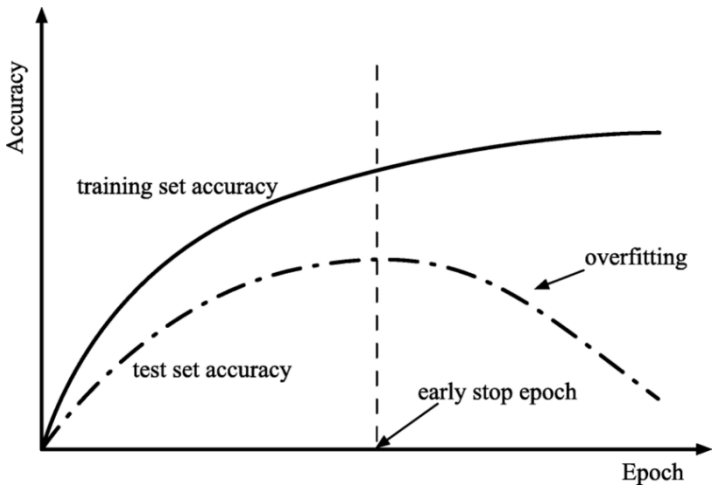
➢ **Optimizers**

Three optimizers were tested to determine the best training strategy:

1. Adam – Adaptive learning rate optimization, learning rate (0.001)
2. SGD – Stochastic Gradient Descent, learning rate (0.01)
3. Adagrad – Adaptive learning with per-parameter learning rates, learning rate (0.01)

➢ **Training Strategy**

- Loss function: categorical_crossentropy
- Metrics: accuracy
- Callbacks:
  - EarlyStopping to prevent overfitting
  - ReduceLROnPlateau to decrease learning rate on plateau
- Class weights applied to handle imbalanced dataset
- Batch size: 128
- Epochs: up to 100 (with early stopping)



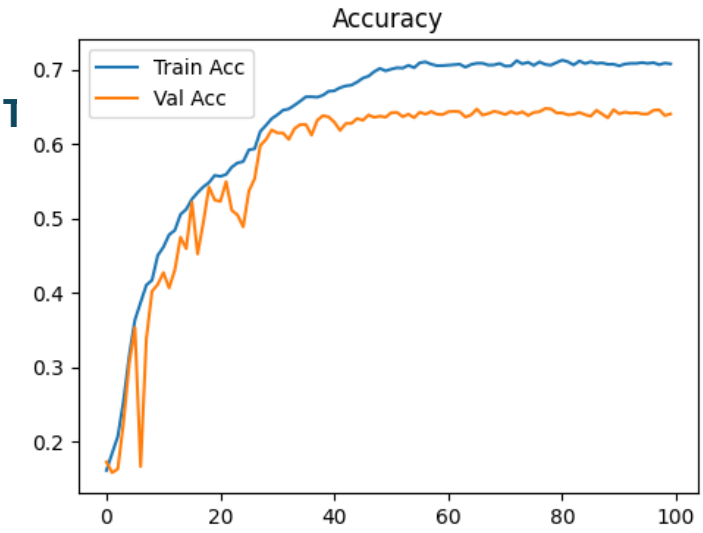
4. Experimental Results

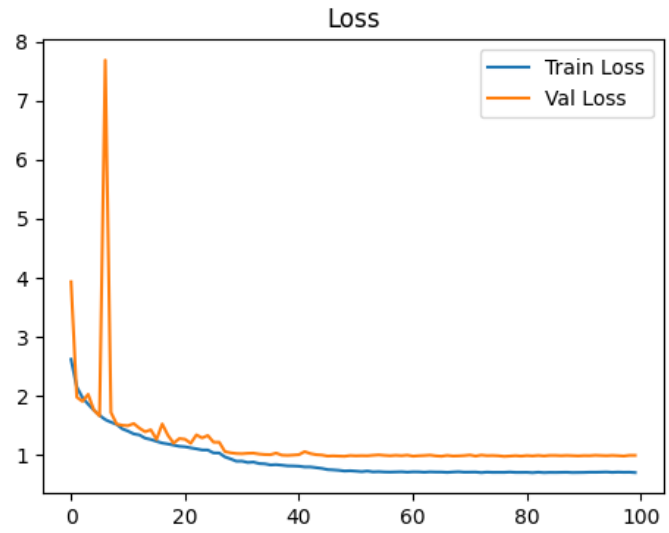
➢ Training and Validation Performance

- Adam optimizer achieved the highest validation accuracy and stable convergence.
- SGD and Adagrad had slower convergence and slightly lower accuracy.

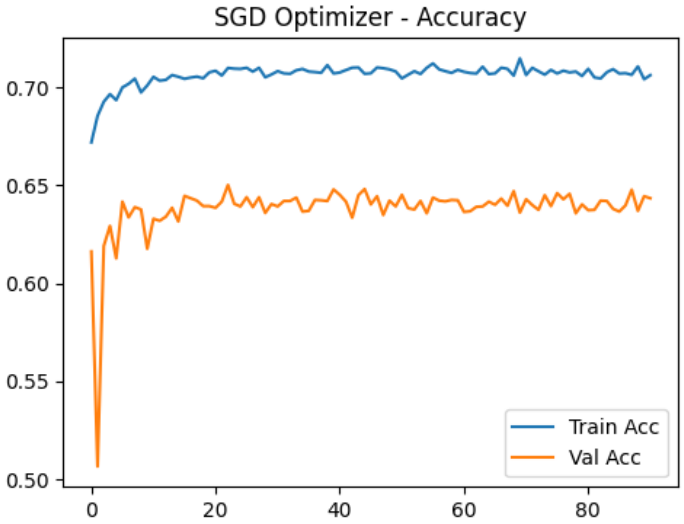
➢ Accuracy & Loss Curves:

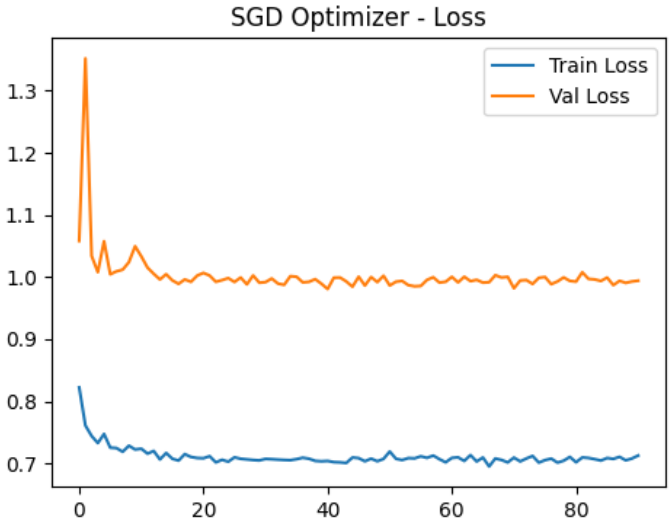
-accuracy &loss for Adam optimizer



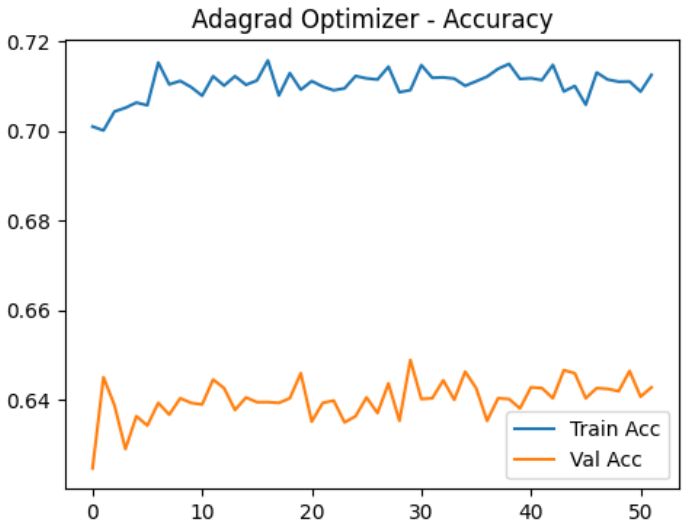


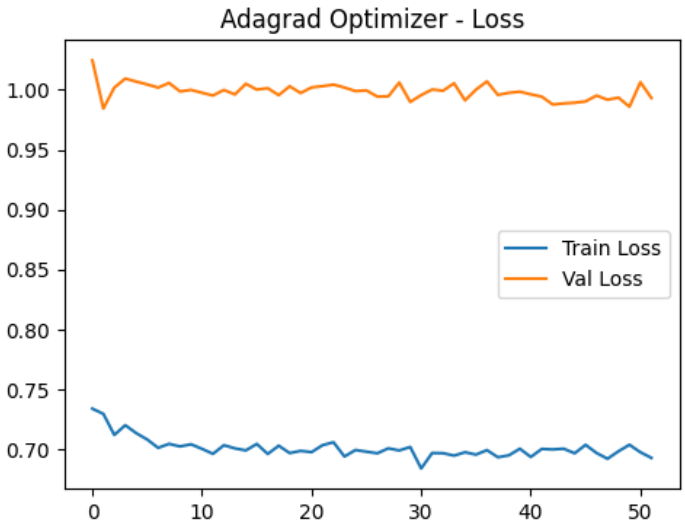
-accuracy &loss for SGD optimizer





-accuracy &loss for Adagrad optimizer





Test Set Evaluation:

```
# Evaluating on test set
test_loss,test_acc=model.evaluate(test_generator)
print("Test Loss:",test_loss)
print("Test Accuracy:",test_acc)

#Evaluating SGD model on test set
test_loss_SGD,test_acc_SGD=model_SGD.evaluate(test_generator)
print("SGD Test Loss:",test_loss_SGD)
print("SGD Test Accuracy:",test_acc_SGD)

# evaluating Adagrad model on test set
test_loss_Adagrad,test_acc_Adagrad=model_Adagrad.evaluate(test_generator)
print("Adagrad Test Loss:",test_loss_Adagrad)
print("Adagrad Test Accuracy:",test_acc_Adagrad)
```

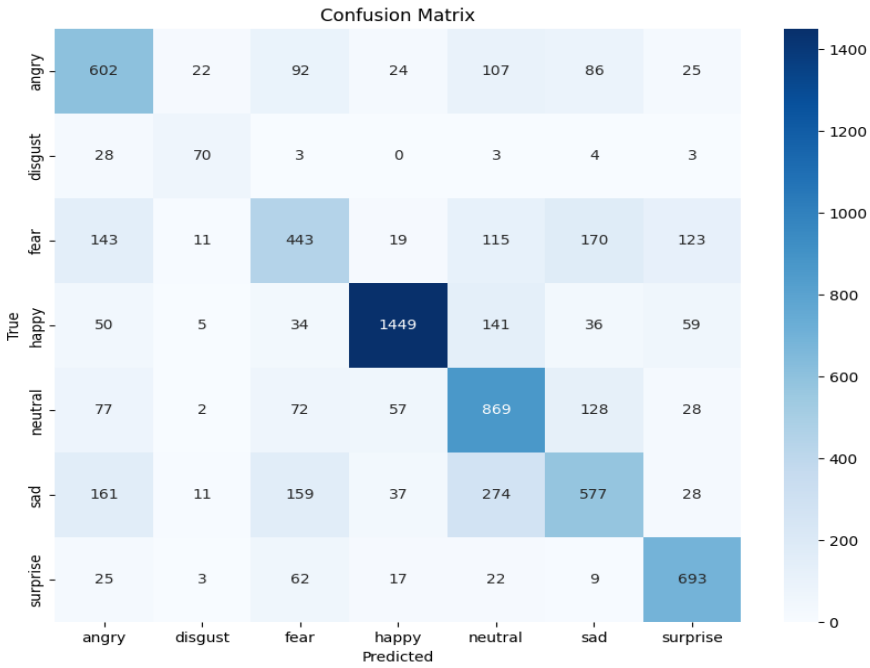
```
57/57 ━━━━━━━━━━━━━━━━━━━━ 27s 480ms/step - accuracy: 0.6306 - loss: 1.0326
Test Loss: 0.9727675914764404
Test Accuracy: 0.6551964282989502
57/57 ━━━━━━━━━━━━━━━━━━━━ 7s 118ms/step - accuracy: 0.6337 - loss: 1.0369
SGD Test Loss: 0.9753552675247192
SGD Test Accuracy: 0.6578434109687805
57/57 ━━━━━━━━━━━━━━━━━━━━ 7s 120ms/step - accuracy: 0.6255 - loss: 1.0456
Adagrad Test Loss: 0.9798542261123657
Adagrad Test Accuracy: 0.6519922018051147
```

- The low accuracy due to firstly because the low accuracy of images and the wrong labeling for image that the image look as fear emotion, but it labeled as sad and so on

```
57/57 ━━━━━━━━━━━━━━━━━━━━ 7s 107ms/step
              precision    recall  f1-score   support

       angry       0.55      0.63      0.59       958
     disgust       0.56      0.63      0.60       111
        fear       0.51      0.43      0.47      1024
       happy       0.90      0.82      0.86      1774
     neutral       0.57      0.70      0.63      1233
         sad       0.57      0.46      0.51      1247
    surprise       0.72      0.83      0.77       831

    accuracy                           0.66      7178
   macro avg       0.63      0.64      0.63      7178
weighted avg       0.66      0.66      0.65      7178

57/57 ━━━━━━━━━━━━━━━━━━━━ 8s 115ms/step
              precision    recall  f1-score   support

       angry       0.55      0.64      0.59       958
     disgust       0.57      0.61      0.59       111
        fear       0.51      0.43      0.46      1024
       happy       0.91      0.82      0.86      1774
     neutral       0.57      0.71      0.63      1233
         sad       0.58      0.46      0.52      1247
    surprise       0.73      0.83      0.78       831

    accuracy                           0.66      7178
   macro avg       0.63      0.64      0.63      7178
weighted avg       0.66      0.66      0.66      7178

57/57 ━━━━━━━━━━━━━━━━━━━━ 7s 113ms/step
              precision    recall  f1-score   support

       angry       0.55      0.63      0.59       958
     disgust       0.55      0.60      0.58       111
        fear       0.55      0.39      0.45      1024
       happy       0.89      0.83      0.86      1774
     neutral       0.54      0.74      0.63      1233
         sad       0.57      0.44      0.50      1247
    surprise       0.72      0.81      0.77       831

    accuracy                           0.65      7178
   macro avg       0.63      0.63      0.62      7178
weighted avg       0.66      0.65      0.65      7178
```

Confusion Matrix

- Visualized to check per class performance.
- Highest accuracy in Happy and surprise emotions.
- Lower accuracy in Fear and Disgust due to
- fewer samples and subtle differences.



Prediction Examples

- Displayed sample correct and incorrect predictions.
- Correct predictions shown in green, incorrect in red.
- This visualization validates model behavior qualitatively

5. Discussion on Model Performance with Justification

- **Adam Optimizer:** Best trade-off between speed and accuracy. Handles sparse gradients well.
- **SGD:** Slower learning; more sensitive to learning rate and requires fine-tuning.
- **Adagrad:** Performs well on sparse data but tends to over-decrease learning rate in longer training.

Observations:

- CNN effectively extracts facial features, but class imbalance impacts minority classes.
- Augmentation improves generalization.
- Deeper networks may benefit further from transfer learning (e.g., pre-trained ResNet).

Justification for Choices:

- Grayscale images reduce complexity and focus on shape/textures.
- Augmentation helps prevent overfitting due to limited FER-2013 dataset.
- Class weights ensure fair contribution of minority classes in loss.

---

6. Appendix with Code

- Git-hub link for code