

Phase 1 Report: Analysis, Design, and Data Preparation

Project: Facial Expression Recognition (FER) System

Course: Artificial Intelligence - K. N. Toosi University of Technology

Instructor: Dr. Pishgoo

Team Members: Soroush Soleimani, Parham Kootzari, Amirhossein Babaee

1. Problem Definition & Scope

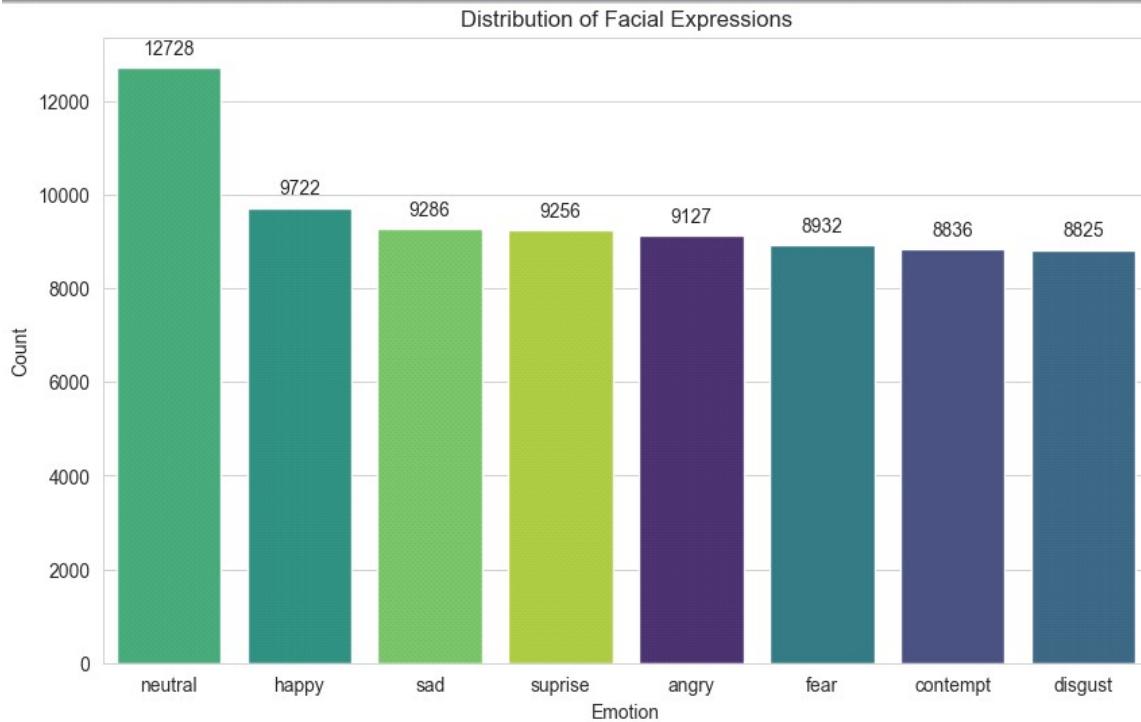
The core objective of this project is to develop a robust Computer Vision pipeline capable of classifying human emotions into seven distinct categories. Facial Expression Recognition (FER) is a pivotal task in Affective Computing, enabling machines to interpret human psychological states. Our system processes grayscale facial images to detect: **Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral.**

2. Dataset Selection & Exploratory Data Analysis (EDA)

We utilized the FER-2013 dataset, a benchmark in the field. Before model design, we conducted an extensive EDA to understand the data's underlying patterns.

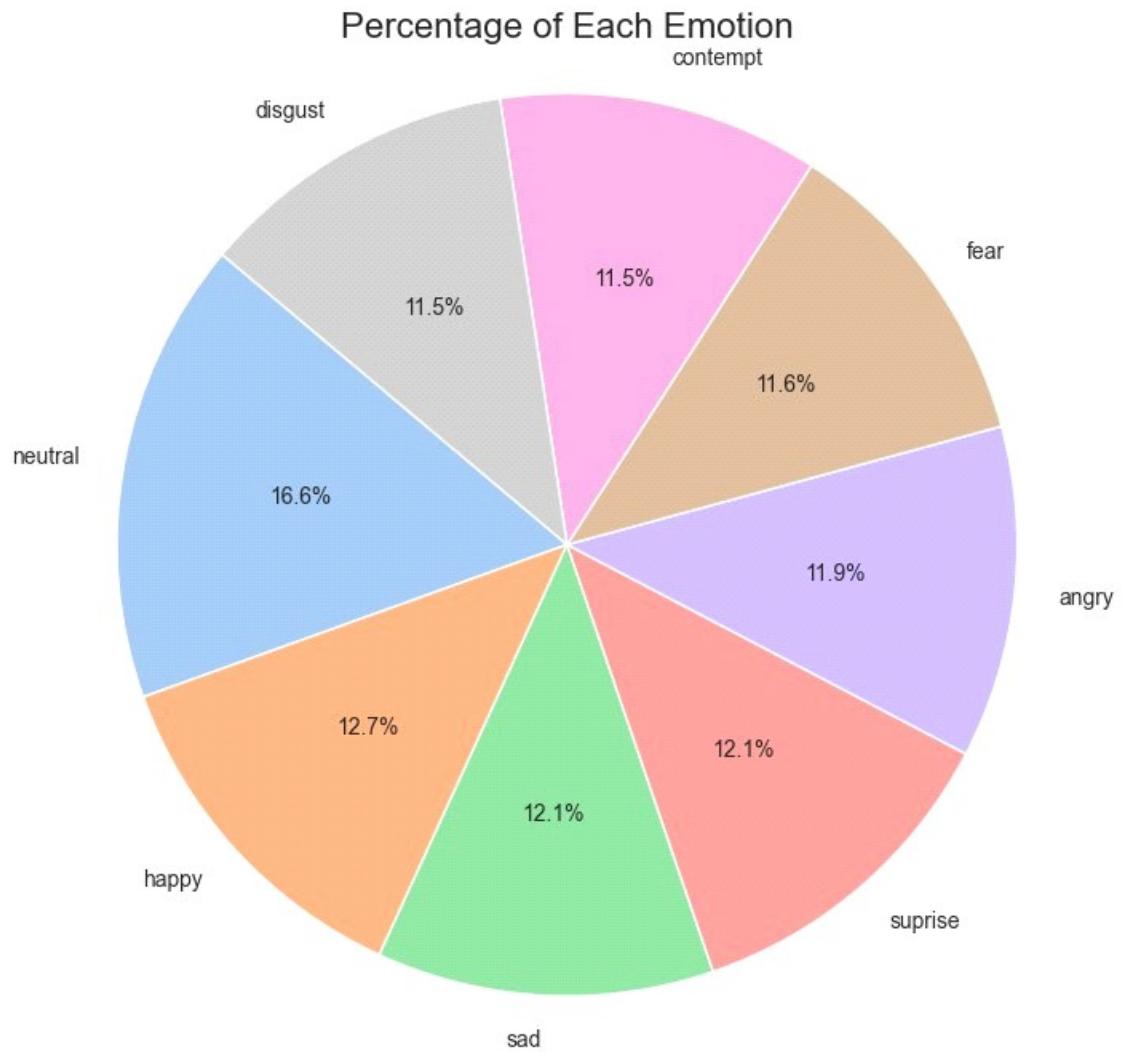
2.1. Class Imbalance Analysis

Our analysis revealed a significant variance in the number of samples per class. For instance, the 'Happy' class has a high density, while 'Disgust' is underrepresented. Recognizing this imbalance is crucial for selecting the appropriate loss function and augmentation strategy.



2.2. Percentage of each emotion

This code counts the occurrences of each emotion and calculates their percentages to check for class imbalance. Its purpose is to ensure the dataset provides enough examples for every category so the model doesn't become biased toward a single expression.



2.3. Visualizing Sample Images

The main objective of this code is visual verification. Before training any AI model, you must confirm that the images are loading correctly and that the labels match the content (e.g., ensuring a file labeled "happy" actually shows a smiling face).



2.4. Visual Data Inspection

To understand the noise levels and variations in head poses/lighting, we visualized samples from each class. This confirmed that the dataset contains real-world challenges like occlusion and low-resolution (48x48) artifacts.



3. Visualizing Feature Separability with t-SNE

- Code Explanation

* **Data Sampling:** It selects a subset of 1,000 images from the dataset. Processing every single image would be computationally heavy, so this sample provides a representative "snapshot" of the data.

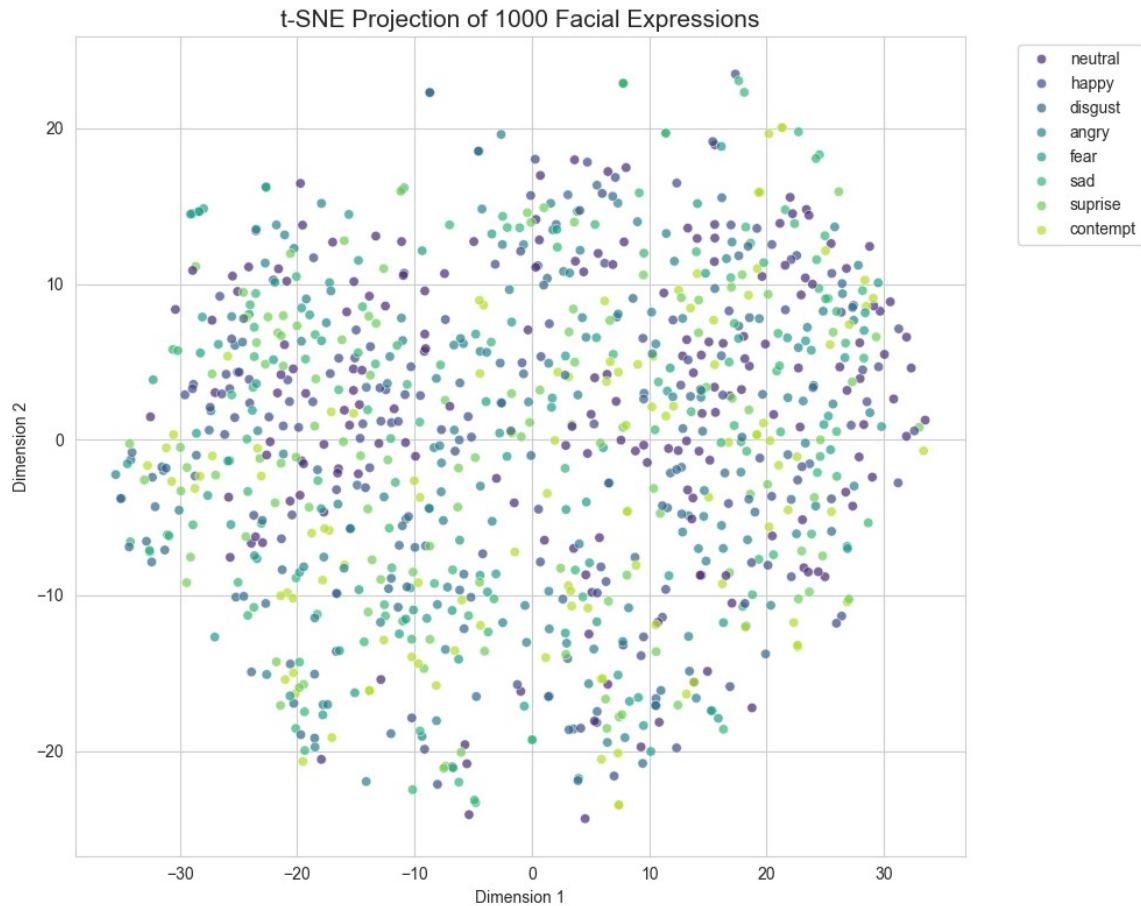
* **Preprocessing:** The code loops through the sampled images, converts them to grayscale, and resizes them to a uniform 48x48 resolution. It then flattens each image into a 1D array of pixels (2,304 features) so the mathematical model can process them.

* **t-SNE Transformation:** It applies the t-SNE algorithm to reduce those 2,304 pixel dimensions down to just two (Dimension 1 and Dimension 2). t-SNE works by keeping similar images close together and pushing dissimilar images apart.

* **Visualization:** Finally, it creates a scatter plot where each point represents an image, colored according to its emotion label.

Purpose

The goal is to see how "separable" the emotions are based on raw pixel values. It helps us understand if certain emotions (like "**Angry**" and "**Disgust**") look very similar to the computer, which would make them harder for the AI to distinguish later.



.4 Edge Detection

Code Explanation

*** Automated Sampling:** The code identifies all unique emotions (labels) in the dataset and selects one random sample image for each category.

*** Dual Visualization:** It creates a grid with two rows:

- **Row 1 (Original):** Displays the raw grayscale facial images to provide a direct look at the data the model will "see."

- **Row 2 (Canny Edge Detection):** Applies the Canny Edge Detection algorithm to the same images. This mathematical operation identifies areas with sharp changes in intensity, effectively highlighting the outlines of facial features like eyes, eyebrows, and mouths.

* **Format:** It uses `plt.subplots` to align them perfectly for a side-by-side comparison across all emotion classes.

Purpose

This section is crucial for understanding feature extraction. Facial expressions are primarily defined by the movement of specific muscles (the "edges" of the mouth or eyes). By visualizing the edges, we can verify if the important structural information of an emotion is preserved after filtering out background noise and lighting variations. It helps confirm that the images are clear enough for a model to recognize specific facial contours.

```

unique_labels = df['label'].unique()

fig, axes = plt.subplots(2, len(unique_labels), figsize=(20, 6))

for i, label in enumerate(unique_labels):
    sample_row = df[df['label'] == label].sample(1).iloc[0]
    path = os.path.join(IMAGE_DIR, sample_row['filename'])

    img = cv2.imread(path, 0)

    if img is not None:
        axes[0, i].imshow(img, cmap='gray')
        axes[0, i].set_title(f"Original: {label}")
        axes[0, i].axis('off')

        edges = cv2.Canny(img, 100, 200)
        axes[1, i].imshow(edges, cmap='gray')
        axes[1, i].set_title(f"Edges: {label}")
        axes[1, i].axis('off')

    plt.tight_layout()
    plt.show()

```

5. Data Preprocessing & Engineering

To transform raw pixels into a format suitable for Deep Learning, we implemented the following pipeline:

- **Normalization:** Pixel intensities were rescaled from [0, 255] to [0, 1] to optimize gradient descent.
- **Feature Cleaning:** We addressed missing values and ensured all images followed the (48, 48, 1) tensor shape.
- **Label Transformation:** Categorical labels were One-Hot encoded to facilitate multi-class cross-entropy calculation.

4. Advanced Data Augmentation

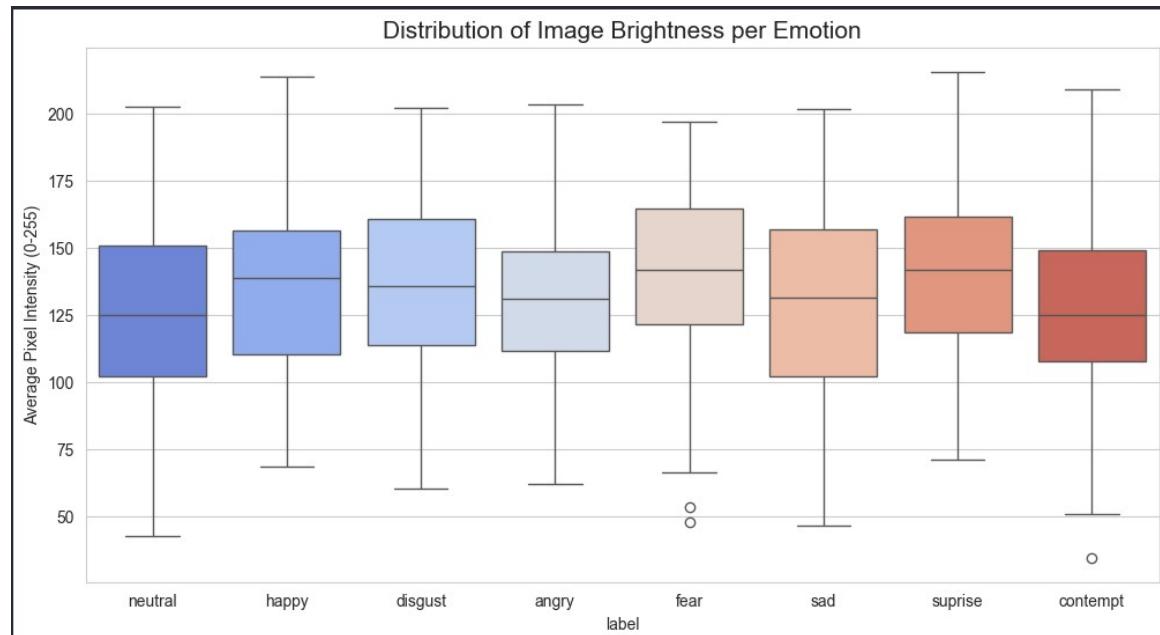
To prevent the model from memorizing the training set (Overfitting), we implemented a real-time augmentation layer. This effectively increases the dataset's diversity by simulating different camera angles and lighting conditions:

- **Geometric Transforms:** Random rotations (15°), width/height shifts, and horizontal flips.
- **Intensity Transforms:** Random zoom and brightness adjustments.

Purpose

The goal is to determine if the lighting conditions are consistent across different emotion categories. In a robust dataset, "Happy" images shouldn't be significantly brighter than "Sad" images just because of the camera settings. If one category is much darker than others, the model might accidentally learn to associate "darkness" with that

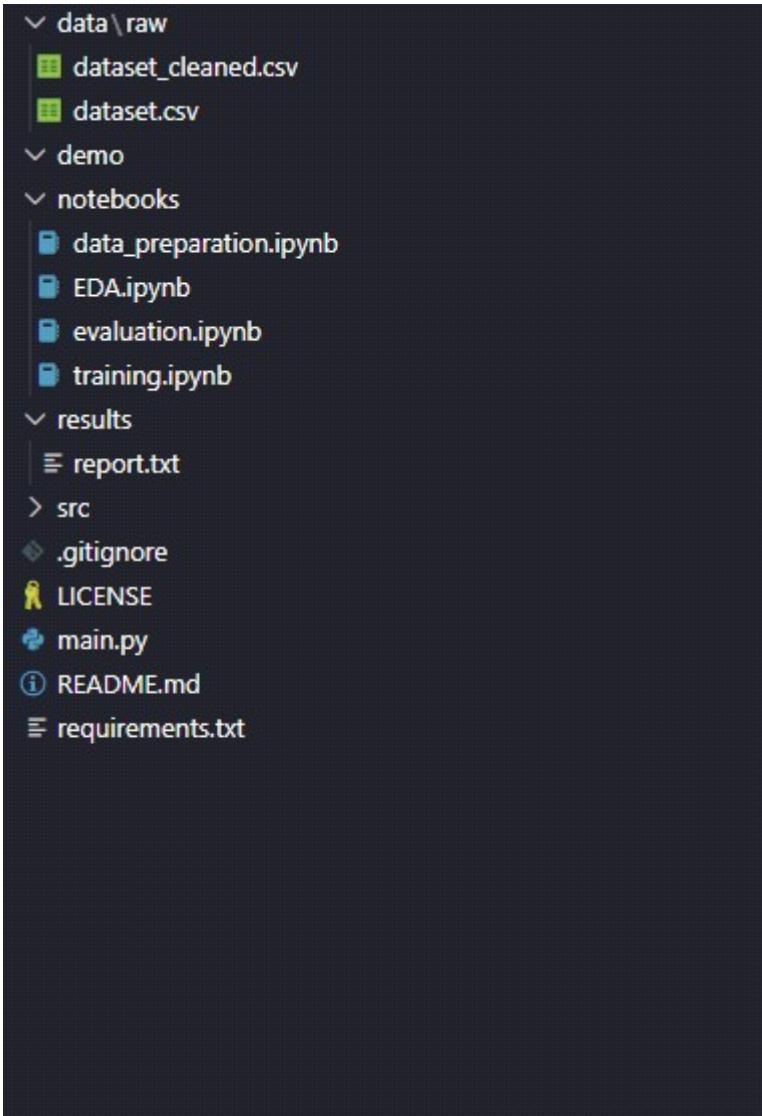
emotion rather than the actual facial features.



5. Architectural Design of the Baseline Model

Following the modularity requirements of the course, we designed a Convolutional Neural Network (CNN). The architecture consists of:

- **Convolutional Blocks:** For hierarchical spatial feature extraction.
- **Batch Normalization:** To stabilize the learning process.
- **Dropout Layers:** To enhance the model's generalization capability.



A screenshot of a file explorer window showing a project directory structure. The root directory contains several subfolders and files:

- data\raw**: Contains `dataset_cleaned.csv` and `dataset.csv`.
- demo**
- notebooks**: Contains `data_preparation.ipynb`, `EDA.ipynb`, `evaluation.ipynb`, and `training.ipynb`.
- results**: Contains `report.txt`.
- src**
- .gitignore**
- LICENSE**
- main.py**
- README.md**
- requirements.txt**

6. GitHub Integration & Version Control

As part of our professional workflow, the project is maintained on GitHub with a modular structure. We used Git for collaborative development, ensuring all changes in preprocessing and model design are tracked through meaningful commits.

Commits on Feb 12, 2026

- report_Phase1_Initial version.pdf
am1383-am committed 7 minutes ago 3b18511
- report_Phase1_Initial version.word
am1383-am committed 8 minutes ago 13cd159

Commits on Feb 12, 2026

- add plot_results.py in utils and change main.py
p-ktz committed 2 days ago 64ff9a2
- add evaluation notebooks and scripts
p-ktz committed 2 days ago d26ca88
- Merge branch 'phase_one' of https://github.com/SoroushSoleimani/Facial-Expression-Recognition into phase_one
p-ktz committed 2 days ago b9cb03e
- add evalator.py in src/evaluation/ and change main.py in src/
p-ktz committed 2 days ago b8727ad

Commits on Feb 11, 2026

- Ignore trained model files and keras artifacts
SoroushSoleimani committed 3 days ago e94197d
- Finalize main pipeline integration
SoroushSoleimani committed 3 days ago aa0c042

- Finalize main pipeline integration
SoroushSoleimani committed 3 days ago aa0c042
- Add training loop with monitoring callbacks
SoroushSoleimani committed 3 days ago e6s09ec
- Configure baseline CNN for 8 emotion classes
SoroushSoleimani committed 3 days ago 0c73e49
- fixing directories code
p-ktz committed 3 days ago 1ca2f07
- Merge branch 'phase_one' of https://github.com/SoroushSoleimani/Facial-Expression-Recognition into phase_one
p-ktz committed 3 days ago f6481e0
- fixing directories code
p-ktz committed 3 days ago ceaed24
- Finalize data loader with augmentation and generators
SoroushSoleimani committed 3 days ago 1883908
- Initialize data loader module
SoroushSoleimani committed 3 days ago 16b76fe
- Configure gitignore to track CSV datasets and exclude raw images
SoroushSoleimani committed 3 days ago 5e7e9ad
- Merge branch 'phase_one' of https://github.com/SoroushSoleimani/Facial-Expression-Recognition into phase_one
SoroushSoleimani committed 3 days ago 27afcc8