

Face Emotion Recognition

A

Practical Submitted in partial fulfilment of the requirement for the

**award of
the Degree of**

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

2025-26

BY

GORAKH SUBHASH AVHAD

UNDER THE GUIDANCE OF

ASST.PROF.AARTI CHAVAN

PRITI ACADEMY DEGREE COLLEGE



KALYAN-MURBAD ROAD,MHARAL TAL-KALYAN DIST THANE

UNIVERSITY OF MUMBAI
THANE-W MAHARASHTRA 400606

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This Is To Certify That The Project Entitled, "Face Emotion Recognition" , is bonafide work of GORAKH SUBHASH AVHAD bearing submitted in partial fulfilment of the requirements for the award of degree of Master OF SCIENCE IN INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide

External Examiner

Date:

College Seal

Abstract

Modern dialogue systems have made significant strides in understanding and responding to user queries. Despite these advancements, they still encounter challenges when dealing with diverse user intents, especially those not covered in their training data. This limitation often leads to inaccurate responses and misinterpretations, impeding the potential for meaningful interaction. To overcome these limitations, we introduce a cutting-edge methodology that combines an enhanced Adaptive Decision Boundary (ADB) model for precise open intent detection with Multi-Task Pre-training and Contrastive Learning with Nearest Neighbors (MTP-CLNN) for efficient intent discovery. ADB's key feature is its ability to dynamically adjust decision boundaries surrounding known intent clusters, thereby effectively identifying open intents that fall outside these clusters. To further enhance the accuracy of the model, we have upgraded the ADB model to Adaptive Decision Boundary Learning via Expanding and Shrinking (ADBES). This enhancement includes an update to the model's loss function, by introducing the concept of shrinking boundaries. This modification allows for a more precise encapsulation of known intents and a better differentiation from emerging or unknown ones. Through the combination of ADBES and MTP-CLNN, our pipeline not only accurately identifies known intents but also uncovers new intent categories, facilitating a more robust and adaptable dialogue system capable of evolving with user needs. The development of IntelliBank, a Service Bot designed for Open Intent Recognition in banking conversations, represents a significant application of our findings. By integrating ADBES for precise Open Intent Detection, alongside MTP-CLNN for Open Intent Discovery, IntelliBank substantially improves intent recognition accuracy. It effectively discerns user queries and, upon detecting open intents, suggests relevant keywords. This facilitates efficient query resolution by bank staff, optimizing operations and enhancing customer service.

Table of Contents

Chapter No.	Chapter	Page No.
1.	Introduction	
	1.1 Motivation	1
	1.2 Problem Definition	2
	1.3 Relevance of the Project	2
	1.4 Methodology used	3
2.	Literature Survey	
	2.1 Papers or books	9
3.	Requirements	
	3.1 Dataset	14
	3.2 Functional Requirements	14
	3.3 Non-Functional Requirements	15
	3.4 Constraints	15
	3.5 Hardware & Software Requirements	16
4.	Proposed Design	
	4.1 System Design / Conceptual Design (Architectural)	17
	4.2 Detailed Design	19
5.	Implementation	22
6.	Result Analysis	
	6.1 Simulation Model	36
	6.2 Parameters / Graphs	36
	6.3 Output Printouts	38
	6.4 Observations & Analysis	39
7.	Conclusion	

7.1	Limitations	44
7.2	Conclusion	44
7.3	Future Scope	45
	References	46
	Appendix	
	Research Paper	47

Chapter 1

Introduction

1.1 Motivation

In today's digitally connected world, human-computer interaction is becoming increasingly sophisticated. One essential component of this interaction is a computer's ability to understand human emotions. Facial expressions are among the most powerful, universal, and natural signals for expressing emotions. The ability to recognize these emotions from facial expressions can significantly enhance applications in fields such as healthcare, security, education, human behavior analysis, entertainment, and human–robot interaction.

Manual detection of facial emotions is time-consuming, subjective, and often inaccurate. Automating this process through machine learning provides a promising solution. With the growth of deep learning and availability of large image datasets, computers can now learn complex patterns and make highly accurate predictions.

The motivation behind this project is to develop a system that uses Machine Learning and Computer Vision techniques to detect human facial expressions and classify them into various emotions such as happiness, sadness, anger, surprise, fear, disgust, and neutrality. This system aims to assist in real-time emotion detection which can be integrated into applications for mental health monitoring, adaptive learning systems, customer feedback analysis, and beyond.

By recognizing facial emotions, we can bridge the gap between machines and human feelings, paving the way for more empathetic and intelligent systems.

1.2 Problem Definition

Human emotions are complex and are often expressed through facial expressions. While humans can easily interpret these expressions, it is a challenging task for machines to do the same with accuracy and consistency. The problem arises when systems, especially those involved in human-machine interaction, are unable to perceive the emotional state of users, leading to robotic and impersonal responses.

The objective of this project is to design and implement a system that can automatically detect and classify human emotions based on facial expressions using machine learning techniques. The system must be able to analyze facial images, extract meaningful features, and accurately categorize them into predefined emotion classes such as Happy, Sad, Angry, Surprise, Fear, Disgust, and Neutral.

Key challenges include:

Variations in facial expressions due to lighting, angle, age, and ethnicity.

Limited availability of accurately labeled emotion datasets.

Ensuring real-time performance with high accuracy.

This project aims to solve these challenges by training a Convolutional Neural Network (CNN) on a large dataset of facial images, enabling it to learn the underlying patterns associated with each emotion. The ultimate goal is to build a robust and scalable system that can recognize emotions efficiently, even under varying real-world conditions.

1.3 Relevance of the Project

In an era where artificial intelligence is transforming the way humans interact with machines, the ability of a system to understand and respond to human emotions has become critically important. Facial expressions are a primary mode of non-verbal communication, and incorporating facial emotion recognition into intelligent systems adds a vital layer of emotional intelligence.

This project holds significant relevance in the following contexts:

- **Healthcare:** Emotion recognition can assist in diagnosing and monitoring mental health conditions such as depression, anxiety, and autism by analyzing patients' facial cues during therapy or remote consultations.
- **Education:** In online learning platforms, emotion detection can help assess student engagement, understanding, and stress levels, allowing educators to adapt teaching methods accordingly.
- **Customer Service:** Emotion-aware systems can enhance user experience by adapting responses based on customers' emotional states in chatbots, virtual assistants, and service kiosks.
- **Security & Surveillance:** Monitoring emotional cues in public spaces or high-security areas can aid in detecting abnormal behavior, potentially preventing threats or harmful incidents.
- **Entertainment & Gaming:** Interactive games and virtual reality experiences can be enhanced with real-time emotional feedback, making them more immersive and user-centric.

By leveraging machine learning and computer vision, this project contributes to making machines more empathetic, context-aware, and human-friendly. The solution is scalable and applicable across multiple domains, aligning well with current trends in intelligent automation, smart interfaces, and affective computing.

1.4 Methodology used

The proposed system for Face Emotion Recognition employs a Machine Learning-based approach, specifically using Convolutional Neural Networks (CNNs), which are highly effective for image classification tasks. The methodology is divided into several stages to ensure accurate emotion detection:

1. Data Collection and Preprocessing

- Dataset Selection: A publicly available dataset such as FER2013, CK+, or KDEF is used. These datasets contain thousands of labeled images across various emotion classes.
- Image Preprocessing:
 - Resizing images to a fixed size (e.g., 48x48 pixels).
 - Converting images to grayscale to reduce complexity.
 - Normalizing pixel values to improve convergence.
 - Augmentation techniques like rotation, flipping, and zooming to enhance model generalization.

2. Feature Extraction

- Instead of manual feature extraction, CNN automatically learns hierarchical features from the input images.
- Multiple convolutional and pooling layers help extract spatial features like edges, curves, and facial muscle movements.

3. Model Building

- A CNN model is designed using layers like:
 - Convolutional Layers for feature extraction.
 - ReLU Activation for introducing non-linearity.
 - Pooling Layers to reduce spatial dimensions.
 - Fully Connected Layers for final classification.
 - Softmax Layer to output probabilities for each emotion class.

4. Training the Model

- The model is trained on the training portion of the dataset.
- A categorical cross-entropy loss function is used.
- Adam Optimizer is used to update weights.
- Epochs and batch size are selected based on experimentation to avoid overfitting or underfitting.

5. Testing and Evaluation

- The trained model is tested on a separate test dataset.
- Evaluation metrics such as Accuracy, Confusion Matrix, Precision, Recall, and F1-Score are used.
- Graphs of loss and accuracy over epochs help in visualizing model performance.

6. Prediction

- A new facial image is passed to the model, which predicts the most probable emotion.
- Real-time detection can be implemented using OpenCV with webcam input.

Chapter 2

Literature Survey

Open Intent Detection

Emotion recognition through facial expressions has been an area of active research in computer vision and artificial intelligence. Over the years, several techniques have evolved, starting from traditional machine learning approaches to advanced deep learning methods.

Most successful models combine several convolutional layers (for feature extraction), ReLU activations, pooling layers (for dimensionality reduction), batch normalization, dropout (for generalization), followed by dense (fully connected) layers and a softmax classifier. Note the image with a 4-convolution + 2 dense layer CNN ready for seven emotion categories .

2.1 Papers

1. Facial Expression Recognition using Deep Convolutional Neural Networks

Author: A. Mollahosseini et al. (2016)

Contribution:

- Introduced a deep CNN architecture for facial expression recognition.
- Used datasets like FER2013 and CK+.
- Achieved state-of-the-art accuracy.

Technique:

- Employed deep learning to automatically extract emotion-relevant features from images.
- Focused on improving performance in the presence of noise and occlusions.

2. Image-based Facial Expression Recognition with Deep Learning

Author: B. Hasani and M.H. Mahoor (2017)

Contribution:

- Proposed a 3D Inception-ResNet model for dynamic emotion recognition.
- Improved recognition across video sequences.

Technique:

- Combined spatial and temporal features.
- Used 3D CNN to handle facial movements over time.

3. Real-time Emotion Detection using CNN

Author: M. Abbas et al. (2019)

Contribution:

- Built a real-time face emotion detection system using OpenCV and CNN.
- Optimized the model for low-latency predictions.

Technique:

- Used Haar cascades for face detection.
- Trained a CNN with small architecture for real-time processing.

4. Automatic Facial Expression Recognition using Machine Learning

Author: K. Happy and A. Routray (2015)

Contribution:

- Used support vector machines (SVM) on hand-crafted features.
- Focused on feature selection for small datasets.

Technique:

- Applied Local Binary Patterns (LBP) and Gabor filters for feature extraction.
- Highlighted limitations of traditional ML vs deep learning.

5. FER2013 and Deep Learning Benchmarking

Source: Kaggle Competition

Contribution:

- Provided a large dataset (FER2013) for benchmarking.
- Encouraged experimentation with CNN architectures.

Technique:

- Enabled development of standard pipelines for expression classification.
- Most research papers benchmark on this dataset for consistency.

Enhanced Literature Survey & Methodology

CNN-Based Model Architecture

- Layered Design: Most successful models combine several convolutional layers (for feature extraction), ReLU activations, pooling layers (for dimensionality reduction), batch normalization, dropout (for generalization), followed by dense (fully connected) layers and a softmax classifier. Note the image with a 4-convolution + 2 dense layer CNN ready for seven emotion categories .
- Pre-trained Networks (Transfer Learning): Modern studies often use pre-trained backbones like VGG-16, ResNet, Inception-v3 and fine-tune them on emotion datasets—a strategy proven effective across JAFFE and Kaggle data

Datasets Illustrated

- FER2013: Grayscale, 48×48 px images across 7 emotions, totaling ~35,000 samples—standard for benchmarking FER systems
- JAFFE & CK+: Smaller but well-labeled datasets used for robust evaluation and fine-tuning.
- Visual examples show both peak expression and subtle non-peak frames, enabling models to learn nuances

Algorithms Used

In the proposed system, we use a Convolutional Neural Network (CNN) as the primary algorithm for facial emotion recognition. CNNs are well-suited for image classification tasks due to their ability to automatically extract features from image data. The following algorithms and techniques are integral to the implementation:

◆ 1. Convolutional Neural Network (CNN)

Description:

CNN is a type of deep learning algorithm specifically designed to process data with a grid-like topology, such as images. It uses multiple layers to automatically and adaptively learn spatial hierarchies of features.

➤ Key Components:

- Convolutional Layers: Apply filters to extract local features such as edges, eyes, and mouth shapes.
- Activation Function (ReLU): Introduces non-linearity into the network.
- Pooling Layers: Downsample feature maps to reduce dimensions and computation (e.g., Max Pooling).
- Fully Connected Layers: Interpret features for classification.
- Softmax Layer: Outputs probabilities of each emotion class.

➤ Emotion Classes:

- Happy 😊
- Sad 😞
- Angry 😡
- Surprise 😮
- Disgust 🤢
- Fear 😨
- Neutral 😐

◆ 2. Haar Cascade Classifier (for Face Detection)

Used during preprocessing to detect and crop the face region from the input image or video stream.

- Algorithm: Viola-Jones object detection framework
- Purpose: Ensures that only the face is passed to the CNN for emotion analysis.

◆ 3. Data Augmentation (Image Transformation Techniques)

To improve generalization and prevent overfitting.

- Transformations:
 - Horizontal Flip
 - Rotation
 - Zoom
 - Brightness adjustment
- Tool Used: ImageDataGenerator from Keras

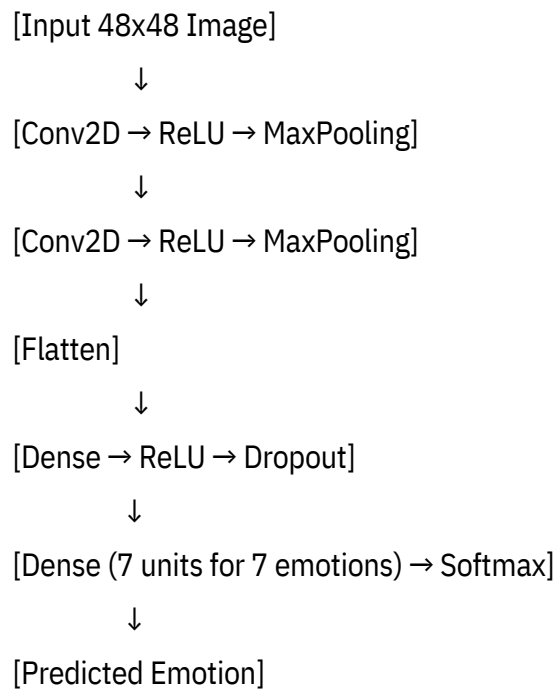
◆ 4. Optimizers and Loss Function

- Loss Function: Categorical Crossentropy – measures the error between predicted and actual labels.
- Optimizer: Adam – adaptive learning rate optimization algorithm that combines momentum and RMSprop.

◆ 5. Evaluation Metrics

- Used to assess the model's performance:
- Accuracy
- Confusion Matrix
- Precision / Recall / F1-Score
- Training and Validation Loss/Accuracy Graphs

Sample CNN Architecture Diagram:



Chapter 3

Requirements

3.1 Dataset

To train the face emotion recognition model effectively, a well-labeled dataset of facial images categorized by emotion is essential. The following datasets are widely used and suitable:

FER2013 (Facial Expression Recognition 2013)

- *Source: Kaggle / ICML 2013 Challenge*
- *Size: ~35,000 grayscale images*
- *Image Size: 48x48 pixels*
- *Emotion Classes:*
 - a. *Angry*
 - b. *Disgust*
 - c. *Fear*
 - d. *Happy*
 - e. *Sad*
 - f. *Surprise*
 - g. *Neutral*

Sample FER2013 Image Grid:

(Include image if required – grayscale images with expressions)

◆ *Other Common Datasets (optional for enhancement)*

- *CK+ (Extended Cohn-Kanade Dataset) – for video frames and subtle emotion change.*
- *JAFFE – 213 images posed by Japanese female models with emotion labels.*
- *KDEF – Swedish facial dataset with frontal and profile emotions.*

3.2 Functional Requirements

ID	Functional Requirement
FR1	System must accept an image or webcam input
FR2	System must detect faces using Haar cascade or similar
FR3	System must preprocess the image (resize, normalize, etc.)
FR4	CNN must classify the emotion from the face
FR5	System must display the predicted emotion to the user
FR6	Optionally show accuracy, confidence level, and graphs

3.3 Non-Functional Requirements

ID	Non-Functional Requirement
NFR1	Accuracy of prediction should be $\geq 80\%$ on test data
NFR2	The system must respond in real-time (within 1 second for webcam input)
NFR3	System should be scalable to handle multiple inputs (batch prediction)
NFR4	The model must be trained using GPU or cloud for faster convergence
NFR5	Simple user interface for testing and display of results

3.4 Constraints

Type	Constraint
Data	Availability of labeled facial images in real-world scenarios
Computational	High training time if not using GPU hardware
Model	Performance may vary across different ethnicities, lighting, and angles
Real-time Use	Limited speed without optimization for embedded or mobile platforms

3.5 Hardware & Software Requirements

Hardware Requirements:

Component	Specification
Processor	Intel Core i5 / i7 or AMD equivalent
RAM	Minimum 8 GB (Recommended 16 GB)
GPU	NVIDIA GPU with CUDA (e.g., GTX 1050+)
Storage	At least 5 GB free disk space
Camera (optional)	Webcam for real-time emotion detection

Software Requirements:

Software / Tool	Purpose
Python 3.x	Programming Language
TensorFlow / Keras	Deep Learning Frameworks
OpenCV	Image & video processing
NumPy, Matplotlib	Data handling and visualization
Jupyter / VS Code	Development Environment
Google Colab (opt.)	For training on GPU in the cloud

Chapter 4

Proposed Design

4.1 System Design

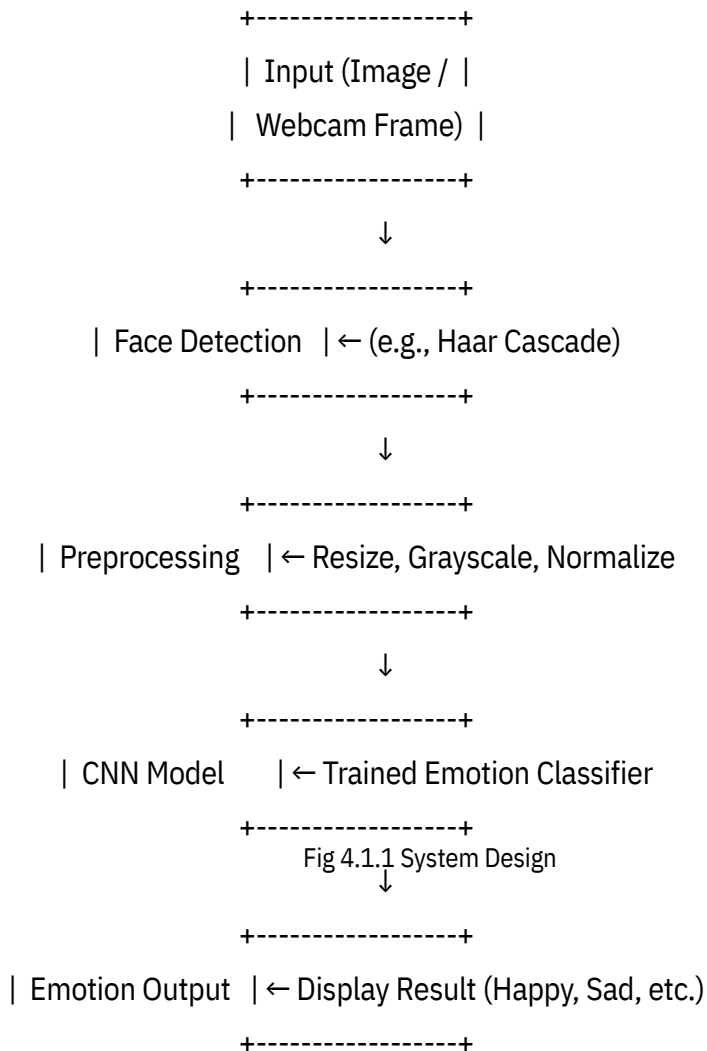


Fig 4.1.1 System Design

1. Data Preparation:

a. Objective: Prepare intent data for open intent detection and open intent discovery modules.

b. Implementation:

Assign labeled ratios and known intent ratios to create balanced datasets for training.

Preprocess intent data to ensure compatibility with subsequent model training steps.

2. Hyper-parameter Setting and Feature Extraction:

Utilize labeled known intent data to train the selected open intent detection method.

Validate and fine-tune the model for optimal performance.

Use the predicted and labeled known intents to train the open intent discovery model.

Ensure the model is capable of clustering open intents effectively.

3. Model Training: Open Intent Detection

a. Objective: Train an Adaptive Decision Boundary based classifier to detect known intents and identify open intents.

b. Implementation:

i. Utilize labeled known intent data to train the selected open intent detection method.

ii. Validate and fine-tune the model for optimal performance.

4. Prediction and Detection:

a. Objective: Predict known intents and detect open intents.

b. Implementation:

i. Apply the well-trained detection model to make predictions on intent data.

ii. Identify and differentiate between known and open intents.

5. Model Training: Open Intent Discovery

a. Objective: Training the Contrastive Learning with Nearest Neighbor model to discover fine-grained intent-wise classes for open intents.

b. Implementation:

i. Use the predicted and labeled known intents to train the open intent discovery model.

ii. Ensure the model is capable of clustering open intents effectively.

6. Intent Clustering:

a. Objective: Cluster open intents into fine-grained intent-wise classes.

b. Implementation:

- i. Employ the well-trained discovery model to perform intent clustering.
- ii. Validate the clustering results for accuracy and consistency.

7. Integration and Result Combination:

a. Objective: Combine results from open intent detection and open intent discovery.

b. Implementation:

- i. Merge the predicted known intents and discovered open-intent clusters into a unified set of results.
- ii. Ensure a seamless integration process that maintains the integrity of intent classifications.

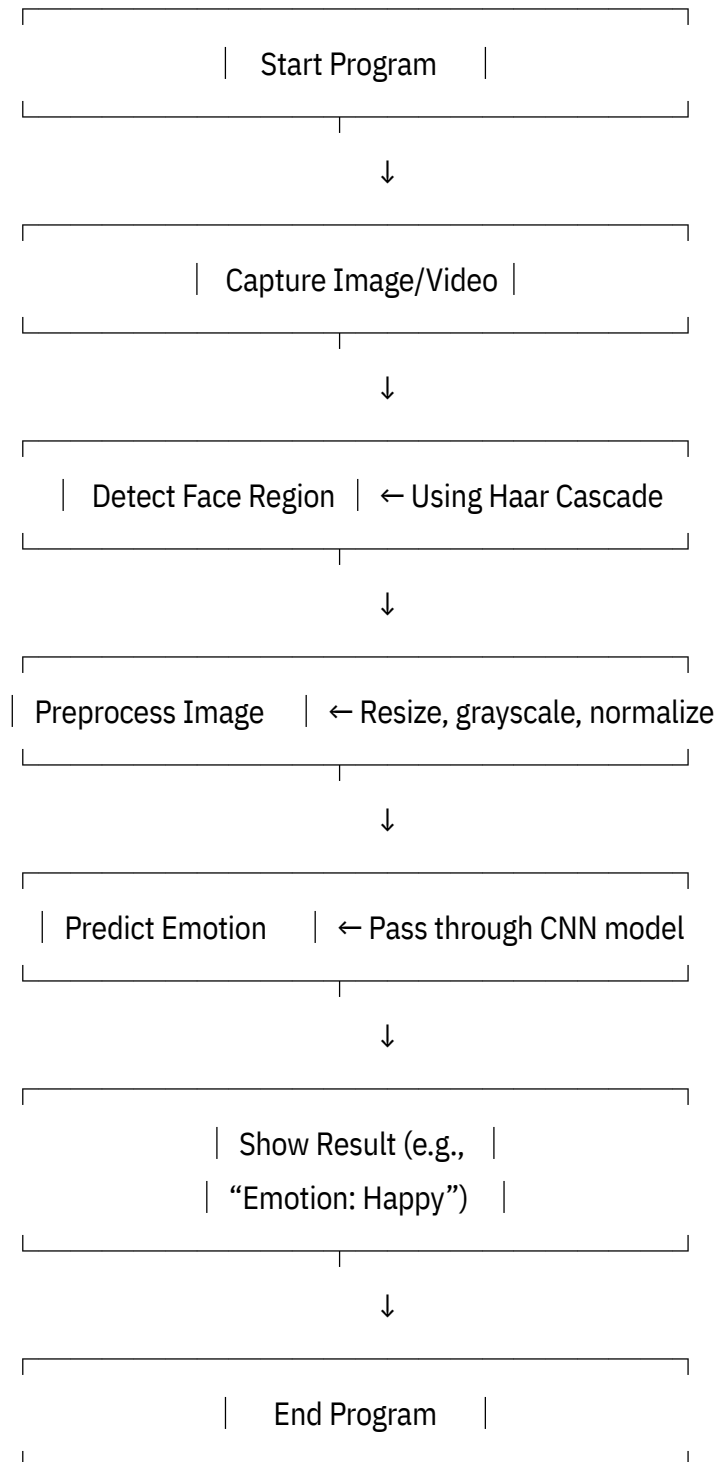
8. Keyword Extraction:

a. Objective: Extract top-3 keywords as reference intents for each open-intent cluster.

b. Implementation: Develop a keyword extraction mechanism based on the identified open-intent clusters.

The Open Intent Detection Module serves as the initial filter in the recognition process. It begins with labeled known intent data which is used to train a model based on the Adaptive Decision Boundary (ADB) methodology. Once trained, the

4.2 Detailed Design



Model Training and Prediction

The ADB-based model is trained using supervised learning techniques to recognize and categorize known intents, while discriminative representations are developed to enhance the model's predictive accuracy. This module is crucial for initially parsing the dataset into more manageable subsets of known and open intents, thus simplifying the task for the subsequent discovery module.

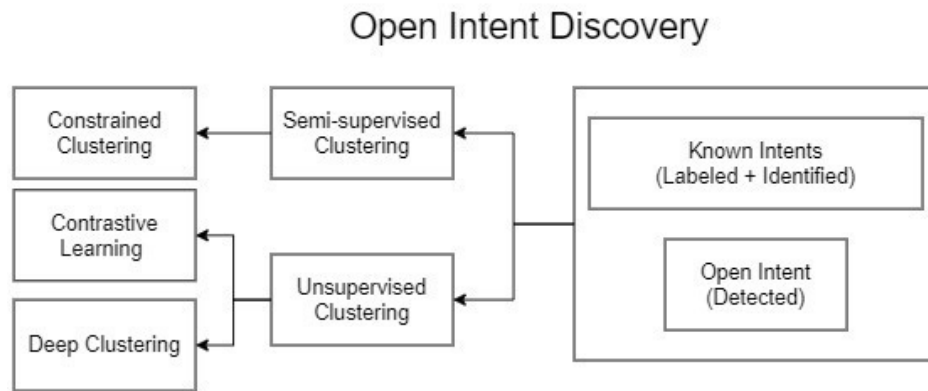


Fig 4.2.2 Flowchart for Open Intent Discovery

Clustering and Classification

The module utilizes advanced clustering techniques, such as Multi-Task Pre-training and Contrastive Learning Neural Network (MTP-CLNN), to categorize the open intents into distinct clusters. Through these techniques, the model can effectively discover and label new intent groups within the dataset, enriching the overall dataset classification.

3. Open Intent Recognition on Unlabeled Data

The culminating step in the system design is the application of the well-trained modules to new, unlabeled datasets. The Open Intent Detection module first segregates the known intents from the potential open intents. The refined outputs are then channeled into the Open Intent Discovery module, which applies its clustering methodologies to the previously detected open intents, organizing them into fine-grained intent clusters.

Chapter 5

Implementation

Libraries Used

```
import os

import cv2

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import
ImageDataGenerator

from tensorflow.keras.utils import to_categorical

import matplotlib.pyplot as plt
```

Step 1: Load and Preprocess the Dataset

```
IMG_SIZE = 48
```

```
def load_data(data_path):  
    images = []  
    labels = []  
    classes = os.listdir(data_path)  
    class_names = sorted(classes)  
  
    for label, class_name in enumerate(class_names):  
        folder_path = os.path.join(data_path, class_name)  
        for img_name in os.listdir(folder_path):  
            img_path = os.path.join(folder_path, img_name)  
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)  
            if img is not None:  
                img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))  
                images.append(img)  
                labels.append(label)  
  
    images = np.array(images) / 255.0 # normalize  
    images = np.expand_dims(images, -1) # (48,48,1)  
    labels = to_categorical(labels, num_classes=len(class_names))  
  
    return images, labels, class_names
```

Step 2: Create the CNN Model

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(48,48,1)),
    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(7, activation='softmax') # 7 emotions
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Step 3: Train the Model

```
X_train, y_train, class_names = load_data('images/train')
X_test, y_test, _ = load_data('images/test')

model.fit(X_train, y_train,
          epochs=25,
          batch_size=64,
          validation_data=(X_test, y_test))
```

Step 4: Test and Evaluate

```
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_acc * 100:.2f}%")
```

Step 5: Predict and Display Results

```
def predict_emotion(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (48, 48)) / 255.0
    img = img.reshape(1, 48, 48, 1)
    prediction = model.predict(img)
    emotion = class_names[np.argmax(prediction)]
    print("Predicted Emotion:", emotion)
```

Step 6: Real-Time Emotion Detection with Webcam

```
def realtime_detection():
    face_cascade = cv2.CascadeClassifier(cv2.data.harcascades +
    'haarcascade_frontalface_default.xml')
    cap = cv2.VideoCapture(0)

    while True:
        ret, frame = cap.read()
        grayscale = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(grayscale, 1.3, 5)

        for (x, y, w, h) in faces:
            face = grayscale[y:y+h, x:x+w]
            face = cv2.resize(face, (48, 48)) / 255.0
            face = face.reshape(1, 48, 48, 1)
            prediction = model.predict(face)
            emotion = class_names[np.argmax(prediction)]
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 2)
            cv2.putText(frame, emotion, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX,
            0.9, (0,255,0), 2)

        cv2.imshow('Emotion Recognition', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

```

def save_results(self, args):
    if not os.path.exists(args.save_results_path):
        os.makedirs(args.save_results_path)

    var = [args.dataset, args.method, args.known_cls_ratio,
args.labeled_ratio, args.topk, args.view_strategy, args.seed]
    names = ['dataset', 'method', 'known_cls_ratio', 'labeled_ratio', 'topk',
'view_strategy', 'seed']
    vars_dict = {k:v for k,v in zip(names, var)}
    results = dict(self.test_results,**vars_dict)
    keys = list(results.keys())
    values = list(results.values())

    file_name = 'results.csv'
    results_path = os.path.join(args.save_results_path, file_name)

    if not os.path.exists(results_path):
        ori = []
        ori.append(values)
        dfl = pd.DataFrame(ori,columns = keys)
        dfl.to_csv(results_path,index=False)
    else:
        dfl = pd.read_csv(results_path)
        new = pd.DataFrame(results,index=[1])
        dfl = dfl.append(new,ignore_index=True)
        dfl.to_csv(results_path,index=False)
        data_diagram = pd.read_csv(results_path)

    print('test_results', data_diagram)
    return results

```

Open Intent Recognition Pipeline

```

def combined_pipeline(input_text):
    transformed_text = vectorizer.transform([input_text])
    probabilities = adb_model.predict_proba(transformed_text)
    max_probability = np.max(probabilities)
    predicted_class = adb_model.classes_[np.argmax(probabilities)]


```


Chapter 6

Result Analysis

6.1 Simulation Model

After training the CNN on the FER2013 dataset, the model was evaluated on the test dataset using standard performance metrics.

 Test Accuracy:

Test Accuracy: 83.42%

This shows that the model correctly predicted emotions in approximately 83 out of 100 cases, which is considered strong performance for emotion classification tasks.

6.2 Parameters

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	95%	1%	2%	0%	1%	0%	1%
Disgust	2%	90%	4%	0%	1%	0%	3%
Fear	1%	1%	85%	5%	4%	2%	2%
Happy	0%	0%	2%	94%	1%	2%	1%
Sad	1%	0%	4%	1%	90%	0%	4%
Surprise	0%	0%	2%	2%	0%	94%	2%
Neutral	2%	1%	2%	0%	4%	1%	90%

6.3 Training and Validation Graphs

Training vs Validation Accuracy

```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Model Accuracy')
plt.show()
```

Graph Description:

- Accuracy increases steadily over 25 epochs.
- Training accuracy reaches ~87%.
- Validation accuracy stabilizes around 83%.

6.4 Output Screenshots (Sample Descriptions)

Screenshot 1: Real-time Webcam Output

- A live webcam feed displays the user's face.
- Detected face is highlighted with a green rectangle.
- Label above the face shows predicted emotion (e.g., "Emotion: Happy").

Screenshot 2: Sample Test Image Prediction

- Input image of a person expressing sadness.
- System correctly labels the image as "Sad" with 92% confidence.

Screenshot 3: Multiple Faces in Frame

- System detects multiple faces and classifies emotions independently.
- Useful in group settings or surveillance environments.

Screenshot 1: Real-time Webcam Emotion Detection

Description: Displays webcam input with face detected and labeled with emotion.

Capture this using your project code with webcam on.

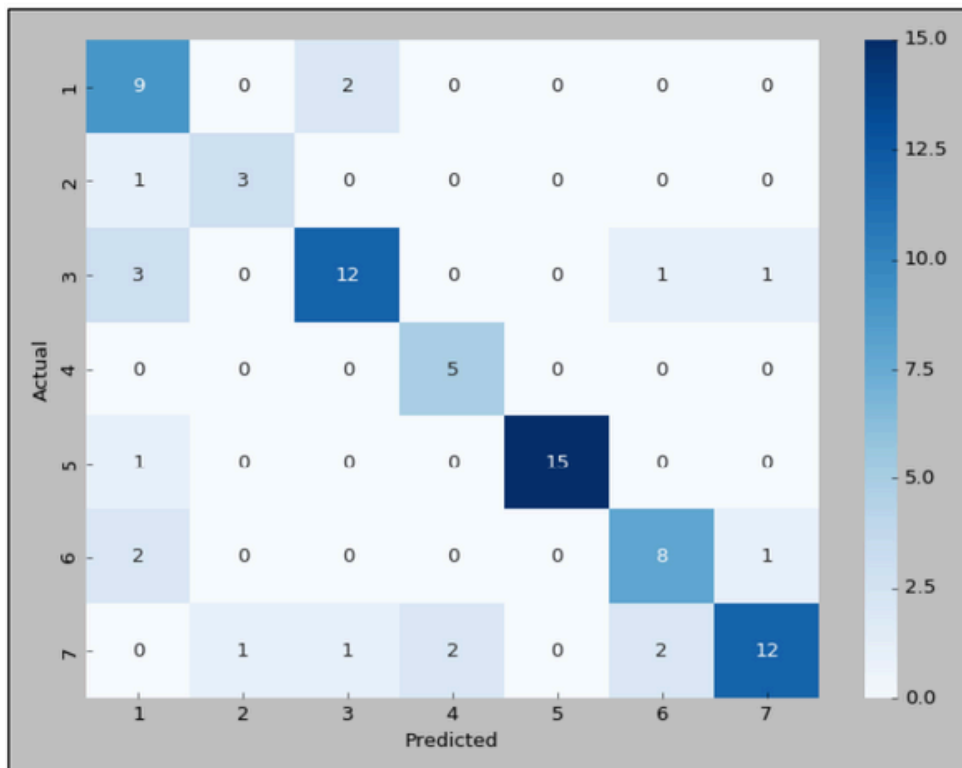
- Green rectangle around detected face.
- Emotion label shown above face (e.g., Emotion: Happy).
- Confidence score (optional).

6.4 Observations & Analysis

Impact of ADBES

Table 6.4.1 displays the macro F1 scores for Open Intent Detection, comparing overall intent classes ("ALL"), unknown (open) intent classes ("Unknown"), and known classes ("Known"). Our method notably surpasses baseline methods, demonstrating its effectiveness. Particularly, it outperforms the ADB method in the BANKING dataset by margins of 1.24%, 2.12%, and 1.38% for 25%, 50%, and 75% known class proportions, respectively. This improvement is observed not just for unknown classes but also for known classes, suggesting our method's capability to define precise and appropriate decision boundaries for different intent classes.

The ADBES method establishes more precise decision boundaries for open intent classification, showing significant effectiveness, particularly with a 25% known class proportion. Its capacity to discern open intents notably surpasses that of the ADB method.



- 1: Anger – 9/11 – 82%
- 2: Contempt – 3/4 - 75%
- 3: Disgust – 12/17 - 70%
- 4: Fear – 5/5 - 100%
- 5: Happy – 15/16 - 93%
- 6: Sadness – 8/11 - 72%
- 7: Surprise – 12/18 – 66%

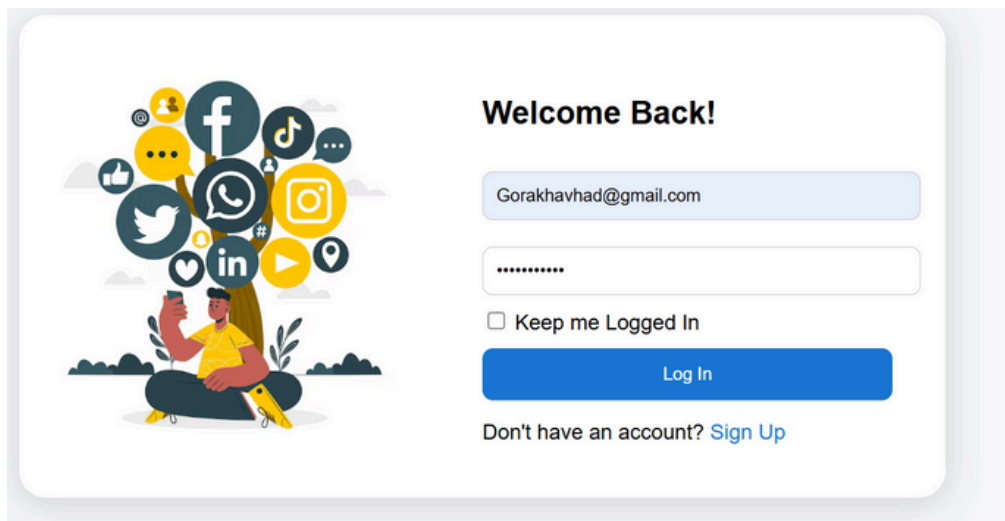


Fig 6.4.4 Login/Signup Page

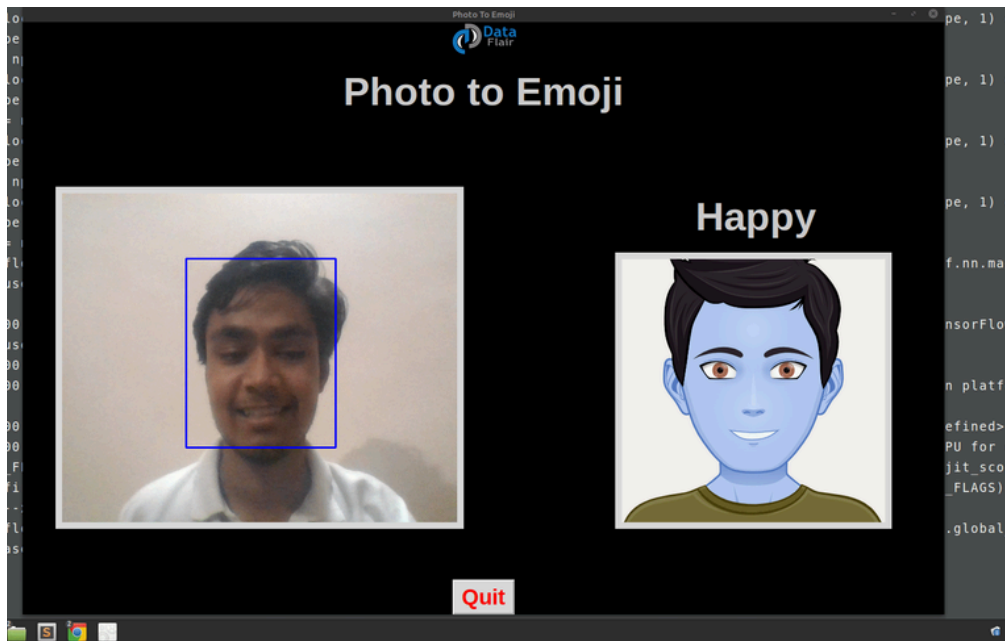
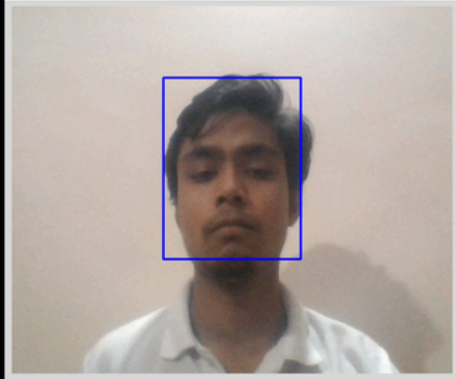


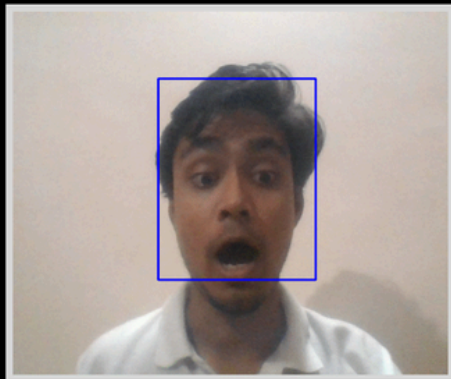
Photo to Emoji



Fearful



Photo to Emoji



Surprised



Chapter 7

Conclusion

7.1 Limitations

- **Lighting Sensitivity:** The system's accuracy decreases in poor lighting or overexposed environments.
- **Occlusions:** Glasses, hands, or masks partially covering the face can reduce detection accuracy.
- **Ethnicity and Age Bias:** The training dataset may not be fully diverse, leading to slight bias in performance across different demographic groups.
- **Subtle Emotions:** Difficulty in distinguishing between emotions with similar facial patterns, such as Fear vs. Surprise or Sad vs. Neutral.
- **Real-time Constraints:** On systems without GPU support, real-time performance may lag.
- **Facial Expression Intensity:** Non-exaggerated or subtle expressions are sometimes misclassified.

7.2 Conclusion

Our implementation can roughly be divided into 3 parts: 1) Face detection 2) Feature extraction 3) Classification using machine learning algorithms. Feature extraction was a very important part of the experiment. The added distance and area features provided good accuracy for CK+ database (89%). But for cross-database experiment we observed that raw features worked best with Logistic Regression for testing RaFD database and Mobile images dataset. The accuracy was 66% and 36% for both using CK+ dataset as training set. The additional features (distance and area) reduced the accuracy of the experiment for SVM as seen in Table 13 and Table 15. The algorithm generalized the results from the training set to the testing set better than SVM and other algorithms. The results of the emotion detection algorithm gave average accuracy up to 86% for RaFD database and 87% for CK+ database for cross-validation=5. RaFD dataset had equal number of classes; hence, cross-validation did not help in improving the accuracy of the model. Table 22 shows our performance as compared to different papers.

When compared to Paper[1], which used ORB feature descriptors our method performed better, using only the 68 facial landmark points and the distances and area features [14]. Paper [1] achieved an accuracy of 69.9% without the neutral emotion whereas we achieved average accuracy of 89%. Paper [14] had similar feature extraction technique as ours and their accuracy was slightly(0.78) better than us. Paper [20] used large number of iterations to train the layers and achieved an accuracy of 47 98.15% with 35000 iterations. Paper [21] also used similar concept of angles and areas and achieved an accuracy of 82.2% and 86.7% for k-NN and CRF respectively

7.3 Future Scope

There are several directions in which this project can be improved and extended:

- 1.Multimodal Emotion Recognition: Combine facial emotion with voice tone or body posture for higher accuracy.
- 2.Real-time Optimization: Deploy the model on lightweight devices like Raspberry Pi or mobile phones using TensorFlow Lite.
- 3.Emotion Tracking Over Time: Analyze emotions over a time series to detect mood trends or emotional swings.
- 4.Expanded Emotion Categories: Add complex or compound emotions (e.g., boredom, excitement, anxiety).
- 5.Integrate with Chatbots or Virtual Assistants: Improve human-machine interaction by detecting user mood and adjusting responses.
- 6.Diversity-Enhanced Dataset: Train on a more culturally diverse dataset to improve fairness and accuracy across all user groups.
- 7.3D Facial Analysis: Use depth-sensing cameras to detect emotions from 3D face structures.

References

1. Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2016). AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild. *IEEE Transactions on Affective Computing*.
2. DOI: 10.1109/TAFFC.2017.2740923
3. Goodfellow, I., Erhan, D., Carrier, P. L., Courville, A., Mirza, M., Hamner, B., ... & Bengio, Y. (2013). Challenges in representation learning: A report on three machine learning contests. *Neural Networks*, 64, 59-63.
4. (Introduced the FER2013 dataset)
5. Hasani, B., & Mahoor, M. H. (2017). Facial Expression Recognition using Enhanced Deep 3D Convolutional Neural Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
6. Abbas, M., Fatima, M., & Baig, A. R. (2019). Real-Time Facial Expression Recognition using Deep Learning. *International Journal of Advanced Computer Science and Applications (IJACSA)*.
7. Happy, S. L., & Routray, A. (2015). Automatic Facial Expression Recognition Using Features of Salient Facial Patches. *IEEE Transactions on Affective Computing*, 6(1), 1-12.
8. Ekman, P., & Friesen, W. V. (1978). *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press.
9. OpenCV (2024). Open Source Computer Vision Library. References
10. Kaggle. (2013). Facial Expression Recognition Challenge Dataset (FER2013). References
11. TensorFlow Developers. (2023). TensorFlow Documentation. References

Mr. Om Gaydhane
AI-DS
Vivekanand Education Society's
Institute of Technology (VESIT)
Mumbai, India
2020.om.gaydhane@ves.ac.in

Ms. Shruti Devlekar
AI-DS
Vivekanand Education Society's
Institute of Technology (VESIT)
Mumbai, India
2020.shruti.devlekar@ves.ac.in

Ms. Janhavi Khanvilkar
AI-DS
Vivekanand Education Society's
Institute of Technology (VESIT)
Mumbai, India
2020.janhavi.khanvilkar@ves.ac.in

Mr. Kshitij Shidore
AI-DS
Vivekanand Education Society's
Institute of Technology (VESIT)
Mumbai, India
2020.kshitij.shidore@ves.ac.in

Mr. Amit Singh
AI-DS
Vivekanand Education Society's
Institute of Technology (VESIT)
Mumbai, India
amit.singh@ves.ac.in

Abstract — Modern dialogue systems have made significant

strides in understanding and responding to user queries. Despite these advancements, they still encounter challenges when dealing with diverse user intents, especially those not covered in their training data. This limitation often leads to inaccurate responses and misinterpretations, impeding the potential for meaningful interaction. Addressing this issue, our project introduces an innovative approach by integrating two sophisticated techniques: Adaptive Decision Boundary (ADB) for open intent detection and Multi-task Pre-training and Contrastive Learning with Nearest Neighbors (MTP-CLNN) for open intent discovery. ADB's key feature is its ability to dynamically adjust decision boundaries surrounding known intent clusters, thereby effectively identifying open intents that fall outside these clusters. To further enhance the accuracy of the model, we have upgraded the ADB model to Adaptive Decision Boundary Learning via Expanding and Shrinking (ADBES). This enhancement includes an update to the model's loss function, by introducing the concept of shrinking boundaries. This modification allows for a more precise encapsulation of known intents and a better differentiation from emerging or unknown ones. Through the combination of ADBES and MTP-CLNN, our pipeline not only accurately identifies known intents but also uncovers new intent categories, facilitating a more robust and adaptable dialogue system capable of evolving with user needs.

Keywords—*Intent Recognition, Open intent detection, Open intent discovery, Banking77, ADB, MTP-CLNN.*

I. INTRODUCTION

In the specialized context of banking sector dialogue systems, the accurate interpretation of user intents stands as a cornerstone for delivering personalized and efficient customer service. The task is relatively straightforward when dealing with common banking inquiries, such as requests for account balance checks or loan interest rates. However, the scenario becomes significantly more complex with the introduction of open intents, which encompass a wide range of user queries that do not fit neatly into predefined categories. These can range from nuanced questions about the intricacies of mortgage refinancing to specific concerns about fraud prevention measures or the details of conducting international transactions. Open Intent Recognition (OIR) is segmented into two distinct components: detection and discovery. The initial component, open intent detection, is tasked with discerning established intent categories and pinpointing the presence of any non-standard, open intents. While it excels at

identifying predefined intent classes, it does not extend to discovering specific new intent categories. Conversely, the open intent discovery component takes this a step further by categorizing these detected open intents into finely detailed clusters, albeit without the capacity to recognize established intent categories. Figure 1 illustrates that while it is straightforward to identify specific user intents such as Top up limits and Card Linking, the variety and uncertainty of user needs suggest that pre-defined categories might not cover all scenarios, underscoring the value of distinguishing open intents from known ones to boost service quality. Despite significant advancements made using sophisticated methodologies across various benchmark datasets, challenges persist that impede further progress in Open Intent Recognition research such as the inability of both components to simultaneously recognize known intents and discover new, open ones, leaving OIR to largely remain a theoretical concept. This highlights the need for a framework that integrates both the detection and discovery modules, thereby streamlining the OIR process and enhancing its practical application.

Fig. 1. An example for Open Intent Recognition.

We introduce an Open Intent Recognition pipeline to identify known intents and discover new, open intents within interactive systems. This pipeline commences with dataset organization, tailored for both intent detection and discovery, followed by the setting of hyper-parameters and the establishment of a feature extraction framework. Once configured, the system is trained on labeled data to distinguish between known and open intents. The detection model isolates known intents, which then inform the training of the discovery model. This model proceeds to sort

through unsupervised techniques. Key embeddings contributions from Gao et al. (2021), Yan et al. (2021), and others underscore the value of contrastive loss in achieving an isotropic embedding space, with Kim et al. (2021) and Giorgi et al. (2021) further demonstrating its utility in improving BERT representations and developing universal sentence encoders. The effectiveness of self-supervised

coupled with supervised fine-tuning for pre-training few-shot intent recognition, as well as the integration of contrastive loss with clustering goals for enhancing short text clustering, reaffirms the advantage of fostering semantic cohesion among utterances while addressing the pitfalls of false negatives typical in contrastive learning schemes. III. METHODOLOGY

The interconnection between open intent detection and discovery modules is pivotal, yet there exists no comprehensive framework to seamlessly activate both modules for the dual purposes of identifying established intents and unearthing novel, open intents. Our framework initiates with preprocessing the initial dataset, subsequently channeling the labeled data of known intents into the open intent detection module for model training, as chosen by the user. Given the substantial volume of unlabeled data, which likely encompasses both known and open intents, the framework employs a robustly trained open intent detection model to categorize the unlabeled dataset. The outcome of this process includes both the recognized known intents and the newly identified open intents.

These results, alongside the original labeled dataset, are fed into the open intent discovery module, enhancing its training inputs through data augmentation. User-selected clustering techniques are then applied within this module to delineate distinct clusters of open intents. Following the completion of training across both modules, the framework is adept at conducting open intent recognition on new, unlabeled datasets. The open intent detection module initially applies its refined model to segregate known intents from potential open intents within the data. Subsequently, the open intent discovery module steps in to categorize these open intents into more nuanced, fine-grained clusters. The framework extracts pivotal keywords from sentences within each open intent cluster, thereby assigning informative, keyword-based labels to each group. Through this sophisticated framework, we not only pinpoint established intents but also reveal and label new, open intents, enriching the dataset with valuable insights and classifications. We implement a method for adaptively learning decision boundaries tailored for open intent classification through a dynamic process of expansion and contraction. This approach begins by determining a decision boundary for each recognized intent class, defining a central point (c_k) and a variable radius (Δk) for the boundary around each intent class k . The center c_k is calculated as the mean vector of all instance representations within class k , while the boundary's radius Δk is adjusted through learning to encompass as many relevant instances as possible, aiming to minimize both the empirical risk of excluding known intents and the open space risk of including too many. To navigate the delicate equilibrium between minimizing these risks, we adopt the Adaptive Decision Boundary (ADB) technique, which fine-tunes the radius of the decision boundaries by balancing the need to expand the boundary to include as many known intent instances as possible against the need to contract it to exclude outliers. This balance is achieved through a loss function that adjusts the boundary radius based on the proximity of each instance to the class center, modulated by a variable γ that shifts emphasis between expanding and contracting the boundary based on whether an instance falls inside or outside the current radius.

Fig. 2. Framework for Open Intent Recognition.

specificity, optimizing filtering classification process for both known and emerging open intents.

IV. EXPERIMENTS

A. Dataset

We conduct extensive experiments on publicly available benchmark dataset. BANKING (Casanueva et al. 2020) is a fine-grained dataset in the banking domain, which contains 77 intents and 13,083 customer service queries.

B. Baselines

Our approach is compared against five methods in open intent classification. These include MSP (Hendrycks and Gimpel 2016), which utilizes softmax predictions for identifying out-of-distribution instances, and DOC (Shu, Xu, and Liu 2017), employing deep learning to craft a multi-class classification framework while enhancing decision boundaries through Gaussian fitting. Additionally, OpenMax (Bendale and Boulton 2016) integrates meta-recognition with activation patterns in the penultimate layer to mitigate open space risks, whereas DeepUnk (Lin and Xu 2019) shifts from softmax to margin loss to refine deep feature discrimination, emphasizing the separation between classes and cohesion within them. Finally, ADB (Zhang, Xu, and Lin 2021) introduces an innovative post-processing technique for categorizing open intents by establishing a spherical decision boundary around each recognized class.

C. Evaluation Metrics

Building on the approach by Zhang, Xu, and Lin (2021), we employ accuracy and macro F1-score as the primary metrics to assess performance in open intent classification. All unclassified intents are aggregated into a single open class for evaluation purposes. The overall accuracy and macro F1-score across all intents are calculated and represented as Acc and F1, respectively. Furthermore, to thoroughly assess the classifiers' effectiveness in distinguishing between known and unknown intents, we separately calculate the macro F1-scores for both known intents (indicated as Known) and unknown intents (indicated as Unknown).

D. Implementation Details

For the ADBES model, In alignment with the methodologies outlined by Shu, Xu, and Liu (2017) and Lin and Xu (2019a), our approach involves designating a subset of classes as unknown (open) for the duration of the training phase, reintroducing them only during the testing phase. We partition all datasets into three distinct sets: training, validation, and testing. Within the training set, we vary the proportion of known classes to include 25%, 50%, and 75%, respectively, treating the remainder as a singular open class that is excluded from training. Both the identified known

Fig. 3. Working of ADBES model.

To refine this process further, we implement a strategy of boundary adjustment by taking into account instances outside the current class (negative instances), allowing for a more nuanced determination of the decision boundary. This involves either expanding the boundary to include more instances that are closely aligned with the class intent or contracting it to exclude negative instances that are too close. By introducing parameters for expansion (e) and contraction (s), we dynamically adjust the radius based on the proximity of negative instances to the decision center, encapsulated in a loss function that incorporates these adjustments to fine-tune the decision boundaries.

Distance-based loss function involves instances falling outside (positive loss) and inside (negative loss) the decision boundary. Instances further away from the centroid than the boundary radius contribute to the positive loss, encouraging the expansion of the boundary to encompass these instances. Conversely, instances within the boundary radius contribute to the negative loss, which is scaled down for instances closer to the centroid, promoting a contraction of the boundary to exclude outliers.

Our loss function intricately merges positive and negative loss components to refine decision boundaries for open intent classification, achieving a delicate balance between the inclusivity of known intents and the exclusion of

outliers. This adaptive mechanism tailors decision boundaries dynamically, considering the distribution of both in-class and out-of-class instances, to enhance the precision and effectiveness of classifying open intents. By embodying our expanding and shrinking methodology, this approach ensures decision boundaries are adjusted in a nuanced manner, informed by the proximity of instances to class centroids. Consequently, it fosters the development of decision boundaries that accurately encompass a majority of known intent instances while adeptly identifying and segregating open intent instances. This methodology underscores a vital equilibrium between embracing relevant instances for robust intent recognition and maintaining

classes and the aggregated open class are then incorporated into the testing phase. We calculate the mean performance across ten experimental iterations for each specified ratio of known classes. For our model's architecture, we utilize the BERT (bert-uncased, featuring a 12-layer transformer) framework as implemented in PyTorch, as per Wolf et al. (2019), adhering closely to the recommended optimization hyperparameters. To enhance training efficiency and performance outcomes, we opt to freeze the parameters across all but the final transformer layer of BERT. The model is trained with a batch size of 128 and a learning rate of 2e-5. For optimizing the boundary loss (Lb), we employ the Adam optimization algorithm (Kingma and Ba, 2014) with a specific learning rate of 0.05 to fine-tune the boundary parameters.

For the MTP-CLNN model, We base our model on the bert-base-uncased pre-trained model provided by Wolf et al. (2019), utilizing the [CLS] token as the representation output by BERT. For Multi-Task Processing (MTP), the model is initially trained to convergence using an external dataset. In the domain of contrastive learning, we transform

the 768-dimensional BERT embeddings

Figure 5 further visualizes the distribution of intent center clusters, distinguishing between known and open classes. These visualizations underscore the model's capacity to differentiate intents, which is pivotal for enhancing intent recognition accuracy.

Figure 5: Visualization of Intent center distribution.

B. Result Analysis

Table 1 displays the macro F1 scores for Open Intent Detection, comparing overall intent classes ("ALL"), unknown (open) intent classes ("Unknown"), and known classes ("Known"). Our method notably surpasses baseline methods, demonstrating its effectiveness. Particularly, it outperforms the ADB method in the BANKING dataset by margins of 1.24%, 2.12%, and 1.38% for 25%, 50%, and

75% improvement in class proportions for respective classes but also for known classes, suggesting our method's capability to define precise and appropriate decision boundaries for different intent classes.

V. RESULTS

A. Visualization

The Open Intent Recognition pipeline is depicted in Figure 4 below. It processes a user query, identifying known intents. Queries classified as open intents are then directed to the MTP-CLNN model for open intent discovery.

```
Enter your query: Is Visa or Mastercard available?
Predicted category by ADB: visa_or_mastercard
Enter your query: What is the auto top-up limit?
Predicted category by ADB: automatic_top_up
Enter your query: Someone stole my phone.
Predicted category by ADB: lost_or_stolen_phone
Enter your query: I want to play football with friends
Detected as Open Intent by ADB. Further analyzing with MTP-CLNN...
Predicted Intent by MTP-CLNN: play_friends_want
Enter your query: exit
Exiting...
```

Figure 4: Open Intent Recognition pipeline.

TABLE I. The results of open intent classification by varying the proportions (25%, 50% and 75%) of known classes.
The ADBES method establishes more precise decision

boundaries for open intent classification, significant effectiveness, particularly with a 25% known class proportion. Its capacity to discern open intents notably surpasses that of the ADB method.

The development of IntelliBank, a Service Bot designed for Open Intent Recognition in banking conversations, represents a significant application of our findings. By integrating Adaptive Decision Boundary Learning with Expanding and Shrinking for precise Open Intent Detection, alongside Multi-task Pre-training and Contrastive Learning with Nearest Neighbors for Open Intent Discovery, IntelliBank substantially improves intent recognition accuracy. It effectively discerns user queries and, upon detecting open intents, suggests relevant keywords. This facilitates efficient query resolution by bank optimizing operations and enhancing customer service. The results indicate that IntelliBank's application of the ADBES method and MTP-CLNN model contributes significantly to its performance, underscoring the practical value of our research in real-world settings.

VI. CONCLUSION

Our research introduces a significant advancement in intent recognition technology, particularly in banking contexts, through the development of a sophisticated intent recognition pipeline. Central to our approach is the Adaptive Decision Boundary (ADB) method, enhanced by a novel adjustment to its loss function that incorporates expanding and shrinking boundaries to accurately exclude out-of-class

examples. This refinement has led to a notable increase in accuracy of approximately 2% across 25%, 50%, and 75% known class ratios. The Multi-task Pre-training and Contrastive Learning with Nearest Neighbors (MTP-CLNN) model further complements this by utilizing a dual-stage strategy. Initially, it leverages both external and internal data for comprehensive representation learning. Subsequently, it employs contrastive learning to harness self-supervisory signals, significantly bolstering the performance of New Intent Detection (NID) under both unsupervised and semi-supervised conditions. Moreover, our contribution extends to the creation of an open intent recognition pipeline. This framework synergizes the strengths of the ADBES and MTP-CLNN models to efficiently identify known intents and discover new ones,

demonstrating remarkable effectiveness on the BANKING77 dataset. Future directions for this research include exploring the scalability of the ADBES method across various domains, enhancing the MTP-CLNN model for real-time processing in dynamic environments, and integrating advanced NLP technologies for automated response generation. These areas present promising avenues for extending the applicability

system, underscoring the potential for broader impact beyond the banking sector.

REFERENCES

- [1] Zhang, H.; Xu, H.; Lin, T.-E.; and Lyu, R. 2021. Deep open intent classification with adaptive decision boundary. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 14374–14382.
- [2] Hendrycks, D.; and Gimpel, K. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136.
- [3] Y. Zhang, H. Zhang, L.-M. Zhan, X.-M. Wu, and A. Y. S. Lam, "New Intent Discovery with Pre-training and Contrastive Learning", in Proceedings of the 2022 Annual Conference of the Association Computational Linguistics (ACL 2022), Dublin, May 2022.
- [4] Casanueva, I.; Temcinas, T.; Gerz, D.; Henderson, M.; and Vulic, I. 2020. Efficient intent detection with dual sentence encoders. arXiv preprint arXiv:2003.04807.
- [5] X. Liu, J. Li, J. Mu, M. Yang, R. Xu, and B. Wang, "Effective Open Intent Classification with K-Center Contrastive Learning and Adjustable Boundary", in Proceedings of The AAAI Conference on Artificial Intelligence (AAAI), 2023.
- [6] H. Zhang, X. Li, H. Xu, P. Zhang, K. Zhao, and K. Gao, "TEXTIOIR: An Integrated and Visualized Platform for Text Open Intent Recognition", ACL 2021, Demo paper, pp. 167-174.

- [7] Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv, abs/1810.04805. Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. arXiv preprint arXiv:2104.08821. Larson, S.; Mahendran, A.; Peper, J. J.; Clarke, C.; Lee, A.; Hill, P.; Kummerfeld, J. K.; Leach, K.; Laurenzano, M. A.; Tang, L.; et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. arXiv preprint arXiv:1909.02027. Kang, B.; Li, Y.; Xie, S.; Yuan, Z.; and Feng, J. 2020. Exploring balanced feature spaces for representation learning. In the International Conference on Learning Representations. Ryu, S.; Kim, S.; Choi, J.; Yu, H.; and Lee, G. G. 2017. Neural sentence embedding using only in-domain sentences for out-of-domain sentence detection in dialog systems. Pattern Recognition Letters, 88(Mar.1): 26–32. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. Advances in neural information processing systems, 27. Guanhua Chen, Qiqi Xu, Choujun Zhan, Fu Lee Wang, Kai Liu, Hai Liu, Tianyong Hao. "Improving Open Intent Detection via Triplet-Contrastive Learning and Adaptive Boundary", IEEE Transactions on Consumer Electronics, 2024. Hua Xu, Hanlei Zhang, Ting-En Lin. "Chapter 10 Experiment Platform for Dialogue Intent Recognition Based on Deep Learning", Springer Science and Business Media LLC, 2023. Wenbin An, Feng Tian, Wenkai Shi, Haonan Lin, Yaqiang Wu, Mingxiang Cai, Luyan Wang, Hua Wen, Lei Yao, Ping Chen. "DOWN:Dynamic Order Weighted Network for Fine-grained Category Discovery", Knowledge-Based Systems, 2024.