Facial Emotion Recognition Using OpenCV

Srinikitha Yendru, Samiksha Somireddygari

ABSTRACT

Facial recognition technology has rapidly evolved, finding extensive applications in security, surveillance, human-computer interaction, and various other domains. This paper presents implementation of a facial emotion recognition system utilizing the FER-2013 dataset and state-of-the-art deep learning techniques. The system leverages Convolutional Neural Networks (CNNs) implemented using Keras and TensorFlow, along with image processing capabilities provided by OpenCV. By employing a robust pipeline for data preprocessing, model training, and evaluation, the proposed system demonstrates effective emotion classification from facial images. The system's performance is validated through comprehensive experiments, highlighting its accuracy and potential for real-world applications in automated emotional analysis.

ACM Reference Format:

Srinikitha Yendru, Samiksha Somireddygari. 2024. Facial Emotion Recognition Using OpenCV. In . ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Facial recognition technology has become a critical tool in various fields, including security, surveillance, virtual reality, and human-computer interaction. The ability to automatically identify and verify individuals from digital images or video frames has significant implications for modern applications. In recent years, advancements in deep learning and computer vision have further enhanced the capabilities of facial recognition systems, enabling more accurate and efficient emotion recognition.

The importance of facial emotion recognition extends beyond mere identification. It has potential applications in areas such as mental health assessment, customer experience enhancement, and personalized user interactions. By analyzing facial expressions, systems can infer emotional states, providing valuable insights for various industries.

This paper focuses on developing a facial emotion recognition system using the FER-2013 dataset, a well-known dataset containing labeled facial images representing different emotional states. The system utilizes a Convolutional Neural Network (CNN) architecture to process and classify these facial images. OpenCV is employed for image preprocessing and manipulation, while Keras and TensorFlow provide the deep learning framework necessary for training and evaluating the model.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The implementation includes several key steps: data preprocessing, including image normalization and augmentation; model architecture design, featuring multiple convolutional layers and dense layers; and evaluation of the model's performance using accuracy metrics and confusion matrices. Through these processes, the system aims to achieve high accuracy in emotion classification and demonstrate its applicability in real-world scenarios where understanding human emotions is crucial.

2 LITERATURE REVIEW

We have gone through several papers, these are the few which impacted us the most:

A novel method is used using geometric facial features for facial emotion recognition based on the demand across various domains. Here the data is gathered based on six emotional states (anger, fear, happiness, surprise, sadness, neutral) captured using two kinect depth sensors and RGB HD camera.We got to know to train a classifier which combines angle and distance features by employing K Nearest Neighbors, which is basically K-NN Algorithm[1].

CNNs and transfer learning has impacted the accuracy and diverse settings in a positive way where models like VGG16, VGG19 and EfficientNetB7 are like benchmarks in this domain. This study introduces a standalone CNN model tailored for real-time sentiment recognition, integrating face detection, emotion classification, and live probabilistic labeling from webcam feeds[2].

OpenCV (Open Source Computer Vision Library) provides a comprehensive framework for building real-time computer vision applications. It offers various functionalities for image processing, video analysis, and machine learning. In the context of facial recognition, OpenCV integrates seamlessly with deep learning frameworks such as TensorFlow and PyTorch, enabling developers to leverage state-of-the-art models for face detection and recognition[4].

Image classification systems typically consist of feature extraction followed by a classification stage. Fasel and Luettin provided a comprehensive overview of analytical feature extractors and neural network approaches for recognizing facial expressions, concluding that both approaches were approximately equally effective at the time of their writing in the early 21st century. However, with the advent of extensive training data and enhanced computational power, the performance of neural network-based models has significantly improved.

Key milestones in this evolution include the work by Krizhevsky and Hinton, who developed a deep neural network resembling the functionality of the human visual cortex, capable of categorizing objects from images using the CIFAR-10 dataset. This model also provided insights into how the network's filters decompose images, enhancing the understanding of neural network operations. The launch of the annual Imagenet challenges in 2010 further propelled advancements in image classification, utilizing large-scale labeled data for improving model accuracy[3].

3 METHODOLOGY

3.1 Data Preparation

The dataset utilized for this facial emotion recognition task is the FER-2013 dataset, which consists of grayscale images of facial expressions along with corresponding emotion labels. The following steps were undertaken to prepare the data:

- **Data Loading:** The dataset is loaded from a CSV file using the pandas library. Each image in the dataset is represented as a string of pixel values, separated by spaces. These strings are converted into a 48x48 pixel matrix using NumPy, transforming each image into a 2D array.
- **Data Conversion:** Each image array is normalized and reshaped into a 48x48x1 tensor, where the third dimension represents the single color channel (grayscale). This format is compatible with the input requirements of the Convolutional Neural Network (CNN).
- **Label Encoding:** The emotion labels are one-hot encoded using pandas' `get_dummies` function. This converts categorical labels into binary vectors, with each vector representing the presence or absence of a particular emotion.
- **Data Splitting:** The dataset is split into three subsets: training, validation, and test sets. Initially, 80% of the data is allocated to the training set. The remaining 20% is divided equally into validation and test sets. This splitting ensures that the model is trained on a majority of the data while validating and testing on separate subsets.
- **Data Standardization:** To standardize the data, the mean is subtracted from each image and then divided by the standard deviation. This process scales the data to have a mean of 0 and a standard deviation of 1, which helps in improving the training stability and convergence speed.
- **Data Reshaping:** After standardization, the images are reshaped into a 4D tensor of shape (number of samples, 48, 48, 1). This reshaping ensures that the data is in the correct format for input into the CNN.

3.2 Model Architecture

We utilized a Convolutional Neural Network (CNN) designed with the Keras Sequential API for image classification tasks. The architecture of the model is detailed below:

- **Initial Layer Configuration:**
  - **Convolutional Layers:** The model begins with a convolutional layer applying 32 filters of size 3×3 to the input images, which have dimensions $48 \times 48 \times 1$. This is followed by Batch Normalization to stabilize and accelerate training. Another convolutional layer with 32 filters is added, along with Batch Normalization. The feature maps are then downsampled using a MaxPooling layer with a pool size of 2×2.
- **Subsequent Blocks:**
  - **Block 2:** The architecture is expanded with 64 filters in two consecutive convolutional layers, each followed by Batch Normalization. MaxPooling is applied to reduce the spatial dimensions of the feature maps.
  - **Block 3:** This block further increases the number of filters to 128, using two convolutional layers with Batch Normalization. The spatial dimensions are reduced by MaxPooling.
  - **Block 4:** The final block incorporates 256 filters in two convolutional layers, with Batch Normalization applied after each convolution. MaxPooling is used to downsample the feature maps.
- **Dense Network:**
  - After extracting features through convolutional layers, the output is flattened into a one-dimensional vector. This vector is then processed by three fully connected (Dense) layers with 256, 128, and 64 neurons, respectively, each using the ReLU activation function.
- **Output Layer:**
  - The final layer is a Dense layer with 7 neurons and a softmax activation function, producing a probability distribution across 7 classes for classification purposes.

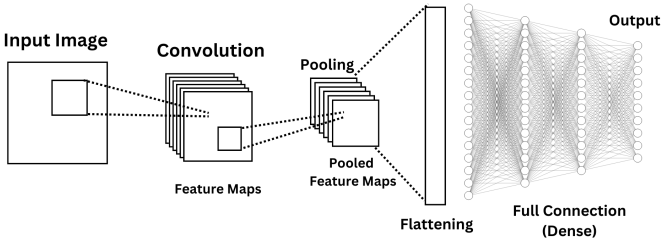

Figure 1: CNN Architecture[7]

This architecture was designed to effectively capture and classify features from input images, leveraging deep convolution layers to extract complex patterns and dense layers to interpret these features for classification.

3.3 Model Training

The model is trained with the following considerations:

- **Data Augmentation:** To enhance model robustness and generalization, various augmentation techniques are applied during training:
  - **Rotation:** Random rotation of images within a range of 20 degrees.
  - **Translation:** Shifts in width and height by up to 20% of the original image dimensions.
  - **Flipping:** Horizontal flipping of images.
  - **Zooming:** Zooming into images by up to 20%.
- **Training Process:** The model is trained for 50 epochs using the Adam optimizer, which adjusts learning rates dynamically. The categorical cross-entropy loss function is used for multi-class classification. The training includes validation data to monitor the model's performance on unseen data.

- **Callbacks:**
  - **CSVLogger:** Logs training and validation metrics to a CSV file for monitoring and analysis.
  - **EarlyStopping:** Stops training if the validation loss does not improve for 5 consecutive epochs, preventing overfitting and restoring the best weights.
  - **ModelCheckpoint:** Saves the model's weights after each epoch to ensure that the best model state is preserved.
- **Batch Size and Epochs:** Training is performed with a batch size of 64 images per step, and the model is trained over 50 epochs. The number of steps per epoch is calculated as the total number of training samples divided by the batch size.

3.4 Model Evaluation

After training, the model is evaluated using the test set to determine its performance. The evaluation includes:

- **Accuracy and Loss Metrics:** Plots of training and validation accuracy and loss over epochs are generated to visualize learning progress and model performance.
- **Confusion Matrix:** A confusion matrix is computed to assess the model's classification performance across different emotion categories. The confusion matrix is visualized using a heatmap, which shows the percentage of correct and incorrect predictions for each emotion class.
- **Prediction Examples:** Random examples from the test set are predicted using the trained model. The predicted emotions are compared with the actual labels to assess the model's performance qualitatively.

4 LIBRARY AND TOOL CHAIN OVERVIEW

The system is implemented using Python with several essential libraries that facilitate various aspects of facial emotion recognition. Each library plays a crucial role in the development and execution of the system:

- **OpenCV (`cv2`)**:
  - **Purpose:** OpenCV, or Open Source Computer Vision Library, is used extensively for real-time image and video processing.
  - **Functions:** It provides functionalities such as real-time video capture from webcams or other video sources, face detection, image transformations (e.g., resizing, cropping), and graphical interface manipulation.
  - **Usage:** In this project, OpenCV is used for capturing live video feeds, detecting faces within these feeds, and performing pre-processing steps such as resizing and normalization of detected faces.
- **NumPy (`np`)**:
  - **Purpose:** NumPy is a fundamental package for numerical computing in Python. It provides support for arrays, matrices, and many mathematical functions.
  - **Functions:** It enables efficient operations on large arrays and matrices, including element-wise operations, mathematical computations, and transformations.
  - **Usage:** NumPy is used for handling image data in the form of arrays, performing normalization, and reshaping images to prepare them for input into the Convolutional Neural Network (CNN).
- **Pandas (`pd`)**:
  - **Purpose:** Pandas is a powerful data manipulation and analysis library. It provides data structures like DataFrames and Series for handling and analyzing structured data.
  - **Functions:** It is used for loading datasets, performing data cleaning, and preprocessing. Pandas also facilitates data splitting and label encoding.
  - **Usage:** In this project, Pandas is used to load the FER-2013 dataset, preprocess the data, and manage data splitting and encoding.
- **Keras (`keras`)**:
  - **Purpose:** Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It allows for easy and fast prototyping of deep learning models.
  - **Functions:** It provides functionalities to define, compile, train, and evaluate neural network models. Keras simplifies the creation of deep learning architectures and the management of model training.
  - **Usage:** Keras is used to build the Convolutional Neural Network (CNN) for emotion recognition. It facilitates the model's creation with various layers, training with data, and evaluation of performance.
- **TensorFlow (`tensorflow`)**:
  - **Purpose:** TensorFlow is an open-source machine learning library developed by Google. It provides a comprehensive ecosystem for developing and training machine learning models.
  - **Functions:** It supports various machine learning algorithms, including deep learning, and provides tools for model optimization and deployment.
  - **Usage:** TensorFlow serves as the backend for Keras, providing the computational infrastructure required for training and evaluating deep learning models. It is used for optimizing model performance and managing computational resources.
- **Matplotlib (`plt`)**:
  - **Purpose:** Matplotlib is a plotting library for creating static, interactive, and animated visualizations in Python.
  - **Functions:** It is used to create graphs and plots, such as training and validation accuracy/loss curves, and other visualizations.
  - **Usage:** In this project, Matplotlib is used to visualize the training process, including plotting the learning curves and performance metrics over epochs.
- **Seaborn (`sns`)**:
  - **Purpose:** Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
  - **Functions:** It simplifies the creation of complex visualizations like heatmaps and pair plots, and enhances the aesthetics of plots.
  - **Usage:** Seaborn is used to generate more sophisticated plots, such as confusion matrices, to provide better insights into model performance.

This comprehensive set of libraries ensures that all aspects of the facial emotion recognition system, from data handling and preprocessing to model training and evaluation, are efficiently managed and executed.

5 OUTPUTS

5.1 Output Obtained through Test Dataset

Below is the output obtained for the test dataset. The model was tested on a set of predefined images with varying facial expressions. The results demonstrate the accuracy and efficiency of our emotion detection algorithm.

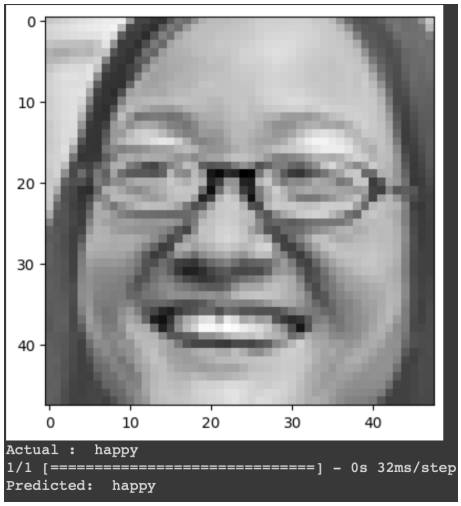

Figure 2: Happy Recognition from Dataset

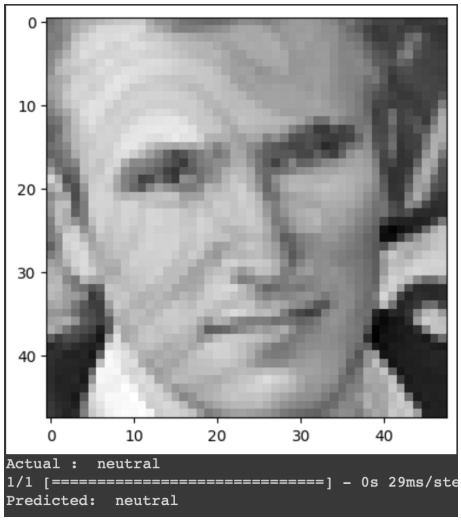

Figure 3: Neutral Recognition from Dataset

s

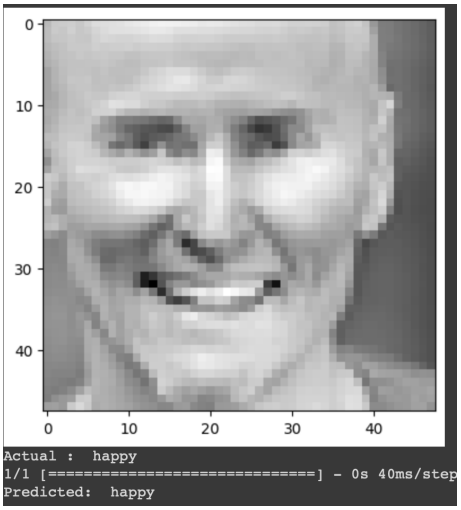

Figure 4: Happy Recognition from Dataset

Figure 5: Sad Recognition from Dataset

5.2 Outputs Obtained from Real-Time live Video

The following output was obtained during live video feed testing. This output represents the real-time emotion detection of faces captured by the camera, showcasing the practical application and performance of our model in dynamic environments.

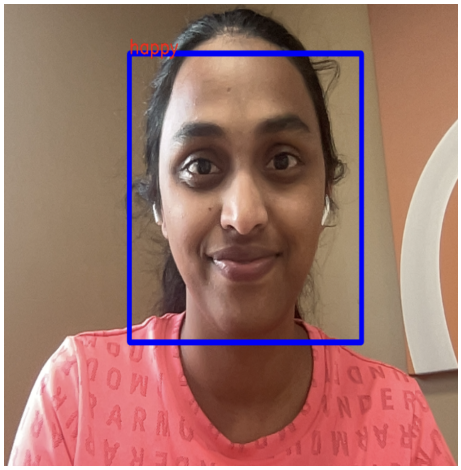

Figure 6: Happy Recognition from Live Video

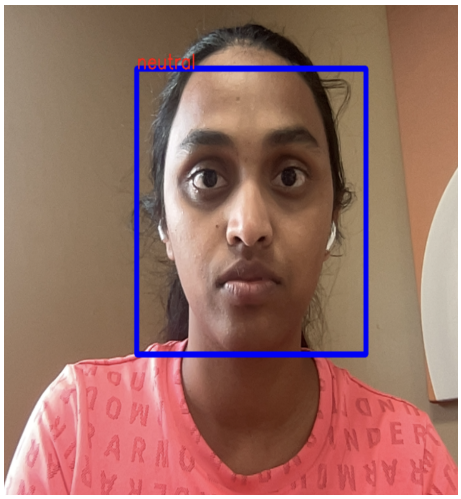

Figure 7: Neutral Recognition from Live Video

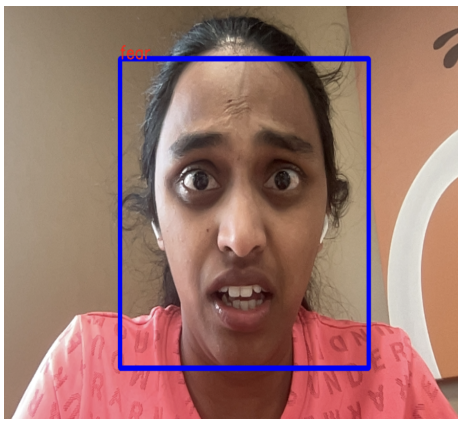

Figure 8: Fear Recognition from Live Video

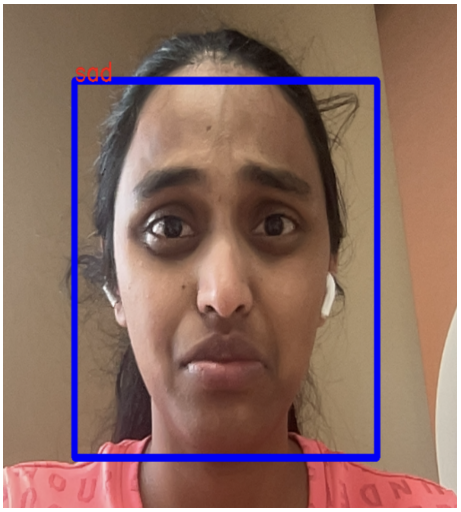

Figure 9: Sad Recognition from Live Video

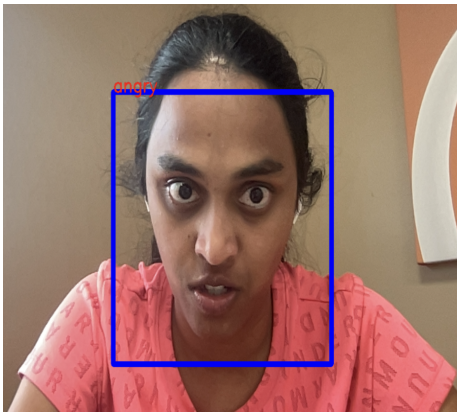

Figure 10: Angry Recognition from Live Video

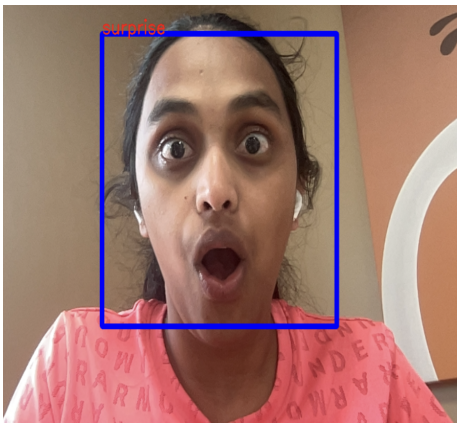

Figure 11: Surprise Recognition from Live Video

6 DISCUSSION

The facial emotion recognition system demonstrated effective performance in recognizing emotions from facial expressions. Future improvements could involve:

- **Handling Variations:** Enhancing the model's ability to handle variations in lighting conditions, facial poses, and subtle expressions.
- **Dataset Expansion:** Incorporating a more diverse dataset with additional emotion categories and varied demographics to improve model robustness.
- **Real-time Performance:** Optimizing the model for real-time applications and integrating it into systems with live video feeds.

7 CONCLUSION

This paper presents a detailed implementation of a facial emotion recognition system using Convolution Neural Networks. The methodology covers data preparation, model architecture, training, and evaluation processes, demonstrating the system's capability in accurate emotion classification from facial images. Future work will focus on refining the model and exploring its applications in real-world scenarios.

REFERENCES

[1] Kahina Amara, Naeem Ramzan, Nouara Achour, Mahmoud Belhocine, Cherif Larbas, Nadia Zenati *Emotion Recognition via Facial Expressions.*
[2] Shubhanjay Pandey, Sonakshi Handoo, Yogesh *Facial Emotion Recognition using Deep Learning*
[3] Akriti Jaiswal, A. Krishnama Raju, Suman Deb *Facial Emotion Detection using Deep Learning*
[4] OpenCV Documentation, https://docs.opencv.org/
[5] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. *FaceNet: A Unified Embedding for Face Recognition and Clustering.* CVPR, 2015.
[6] Parkhi, Omkar M., Andrea Vedaldi, and Andrew Zisserman. *Deep Face Recognition.* BMVC, 2015.
[7] *https://www.pycodemates.com/2023/06/introduction-to-convolutional-neural-networks.html*