KULLIYYAH OF INFORMATION AND COMMUNICATION TECHNOLOGY

CSCI 2304 INTELLIGENT SYSTEMS

SEMESTER 2, 2024/2025

SECTION 4

FACE DETECTION & EMOTION CLASSIFICATION FOR IMPROVING CUSTOMER FEEDBACK USING DEEP LEARNING

GROUP: ARTIFICIAL STUPIDITY

NAME	MATRIC NUMBER
MUHAMMAD AMIRUL HAZIQ BIN MUHAMAD HASMAHADI	2319959
MUHAMMAD UMAIR ASYRAAF BIN SAMSURI	2317901
YUSUF MOHAMMAD YUNUS	2314467
SUHAIB ADNAAN	2317573
ALNATSHEH HUTHIFA M J	2229009

Instructed by:

DR. AMIR ‘AATIEFF BIN AMIR HUSSIN

Table of Contents

Abstract

This project uses deep learning to recognize emotions from people's faces in images. We trained a Convolutional Neural Network (CNN) using the FER2013 dataset, which includes thousands of labeled facial expression images. The model can identify seven emotions: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise. To find faces in user-uploaded images, we used a face detection method called MTCNN (Multi-task Cascaded Convolutional Networks). Once the faces were found, they were passed to our trained CNN model to classify the emotion.

Some challenges we faced included dealing with blurry or low-quality images, detecting small or angled faces, and improving the model's accuracy. Our final CNN model reached an accuracy of 62.97%, which is decent considering the highest possible accuracy one can achieve using this dataset is around 71%. Finally, we deployed our model along with the MTCNN model to perform live face detection and emotion classification using a simple Python script.

Introduction

Understanding how people feel is important for many businesses, especially when trying to improve their products and services. Often, customers may not say exactly how they feel, but their facial expressions and body language can give useful clues. This project aims to help businesses better understand customer emotions by using deep learning to recognize facial expressions from images. By analyzing emotions, businesses could gather helpful feedback without needing direct comments from customers.

We chose to use the FER2013 dataset to train a CNN model because it is publicly available, well-labeled, and contains a large number of facial expression images, with 35,887 grayscale images of faces, where each image has a 48x48 pixel resolution. Although the image quality is not very high, we believed it was good enough to train a model that could still learn useful patterns in facial expressions.

Our goal with this project was to explore how deep learning, especially Convolutional Neural Networks, can be used to automatically detect emotions in faces. We also wanted to combine this with a face detection method so that the program can work on real images where people might appear in different positions or backgrounds. This helped us learn how different AI tools can work together to solve a real-world problem. Our implementation and experiments were conducted using Google Colab, and the full code is accessible at the following link:
https://colab.research.google.com/drive/1OvA5j66II0X3FVnja7aiAQ8j5Y6vj3uo?usp=sharing

The demonstration of our live face detection and emotion classification deployment can be viewed in the link below:
https://youtu.be/oHWkWC97bl0

Methodology / Experimental Setup

Literature Review

Emotion recognition using facial expressions is a well-studied problem in the field of computer vision. Convolutional Neural Networks (CNNs) have become a popular method for such tasks due to their ability to automatically extract important features from images (He et al., 2015). CNNs perform particularly well in classification tasks involving visual patterns, such as detecting emotions from faces.

For face detection, we used MTCNN (Multi-task Cascaded Convolutional Networks), which has shown a good balance between speed and accuracy. MTCNN detects faces using three neural network stages that refine the results progressively (Zhang et al., 2016). Compared to older methods like Haar Cascades, which are fast but often inaccurate, MTCNN performs better on faces with different poses, lighting conditions, and sizes.

Several other approaches exist for emotion detection, including traditional machine learning models like Support Vector Machines (SVMs) or handcrafted features like Local Binary Patterns (LBP), but CNNs generally outperform these methods in terms of accuracy (Mollahosseini et al., 2016).

Implementation

The project was implemented in Python using Google Colab. The TensorFlow and Keras libraries were used to build and train the CNN, and the mtcnn library was used for face detection. Finally, using cv2 we deployed a small and simple program to detect faces and classify emotions in real-time.

Dataset and Preprocessing

We used the FER2013 dataset, which consists of grayscale facial images of size 48x48 pixels labeled with seven emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. The images were stored in separate folders for each emotion. During preprocessing, each image was:

- Resized to 48x48 pixels.
- Converted to grayscale if necessary.
- Normalized to scale pixel values between 0 and 1.
- Converted into NumPy arrays for input into the CNN model.

CNN Model Training

The core of our system is a Convolutional Neural Network (CNN) trained to classify facial expressions into seven emotion categories. The CNN was designed using TensorFlow and Keras and consisted of the following layers:

- Input layer: Accepts 48x48 grayscale images.
- Convolutional layers: Multiple layers using 3x3 filters with ReLU activation, responsible for learning visual features such as edges, textures, and shapes from facial expressions.
- Max-pooling layers: Reduce the spatial dimensions of feature maps, keeping only the most important information and reducing computation.
- Dropout layers: Used to randomly disable certain neurons during training, which helps prevent overfitting and improves generalization.
- Fully connected (dense) layer: Interprets the extracted features to predict emotion classes.
- Output layer: Uses a softmax activation function to produce probabilities for each of the 7 emotion classes.

The model was compiled using the categorical cross-entropy loss function, suitable for multi-class classification, and the Adam optimizer, which adjusts learning rates automatically for faster convergence. Below is the model architecture generated from the 'model.summary()' command:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 256)	2,560
batch_normalization (BatchNormalization)	(None, 46, 46, 256)	1,024
activation (Activation)	(None, 46, 46, 256)	0
conv2d_1 (Conv2D)	(None, 46, 46, 256)	590,080
batch_normalization_1 (BatchNormalization)	(None, 46, 46, 256)	1,024
activation_1 (Activation)	(None, 46, 46, 256)	0
dropout (Dropout)	(None, 46, 46, 256)	0
max_pooling2d (MaxPooling2D)	(None, 23, 23, 256)	0
conv2d_2 (Conv2D)	(None, 23, 23, 128)	295,040
batch_normalization_2 (BatchNormalization)	(None, 23, 23, 128)	512

activation_2 (Activation)	(None, 23, 23, 128)	0
conv2d_3 (Conv2D)	(None, 23, 23, 128)	147,584
batch_normalization_3 (BatchNormalization)	(None, 23, 23, 128)	512
activation_3 (Activation)	(None, 23, 23, 128)	0
dropout_1 (Dropout)	(None, 23, 23, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 11, 11, 128)	0
conv2d_4 (Conv2D)	(None, 11, 11, 64)	73,792
batch_normalization_4 (BatchNormalization)	(None, 11, 11, 64)	256
activation_4 (Activation)	(None, 11, 11, 64)	0
conv2d_5 (Conv2D)	(None, 11, 11, 64)	36,928
batch_normalization_5 (BatchNormalization)	(None, 11, 11, 64)	256
activation_5 (Activation)	(None, 11, 11, 64)	0
dropout_2 (Dropout)	(None, 11, 11, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 512)	819,712
batch_normalization_6 (BatchNormalization)	(None, 512)	2,048
activation_6 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131,328
batch_normalization_7 (BatchNormalization)	(None, 256)	1,024
activation_7 (Activation)	(None, 256)	0
dropout_4 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32,896

batch_normalization_8 (BatchNormalization)	(None, 128)	512
activation_8 (Activation)	(None, 128)	0
dropout_5 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 7)	903

Total params: 6,406,807 (24.44 MB)

Trainable params: 2,134,407 (8.14 MB)

Non-trainable params: 3,584 (14.00 KB)

Optimizer params: 4,268,816 (16.28 MB)

We used early stopping during training to monitor validation loss and prevent overfitting, and the dataset was shuffled and split into training and validation sets. The final trained model achieved an accuracy of 62.97%, which is considered reasonable given the poor image quality and small variability in the dataset.

Face Detection and Testing with MTCNN

To apply the trained model to real-world images, we integrated MTCNN (Multi-task Cascaded Convolutional Neural Network) for face detection. MTCNN is well-suited for detecting faces in images where people may appear at different angles, with varying lighting or occlusion, and in crowded scenes. MTCNN's working consists of three stages:

- Proposal Network (P-Net): Quickly scans the image to propose candidate face regions.
- Refine Network (R-Net): Filters out false positives and refines the bounding boxes.
- Output Network (O-Net): Further improves bounding box predictions and detects facial landmarks (eyes, nose, mouth, etc.).

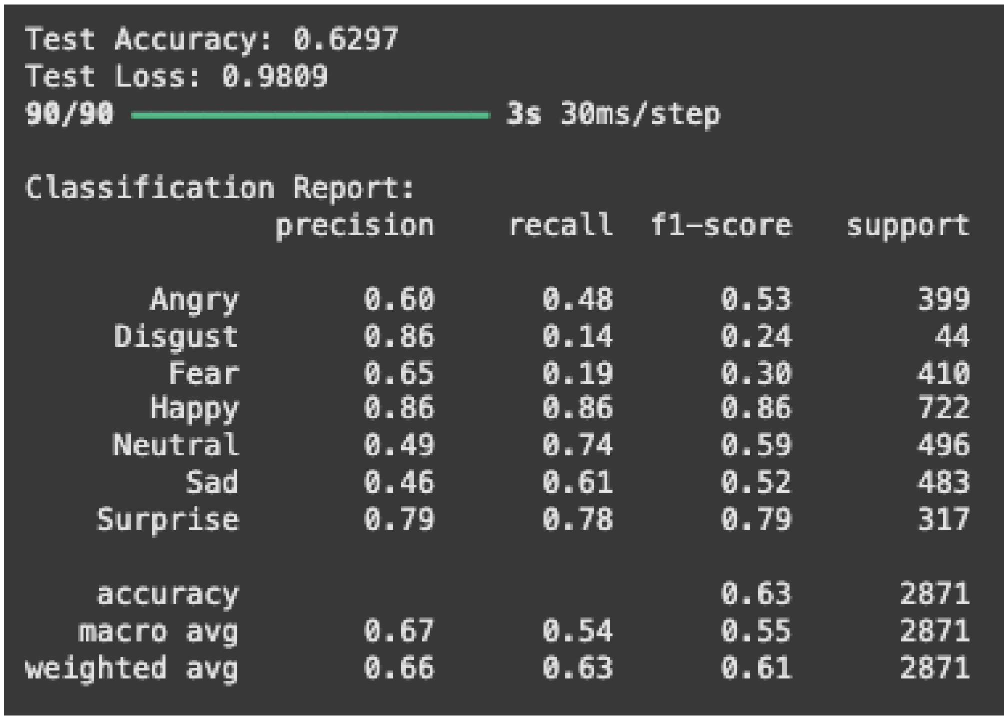
This cascade approach allows MTCNN to balance speed and accuracy, making it better than older methods like Haar Cascades, which rely on hand-engineered features and often fail in more complex images (e.g., tilted faces or shadows). MTCNN is also more efficient than larger models like Faster R-CNN, which are more accurate but slower and not ideal for lightweight applications like ours.

Results & Discussion

Model Performance

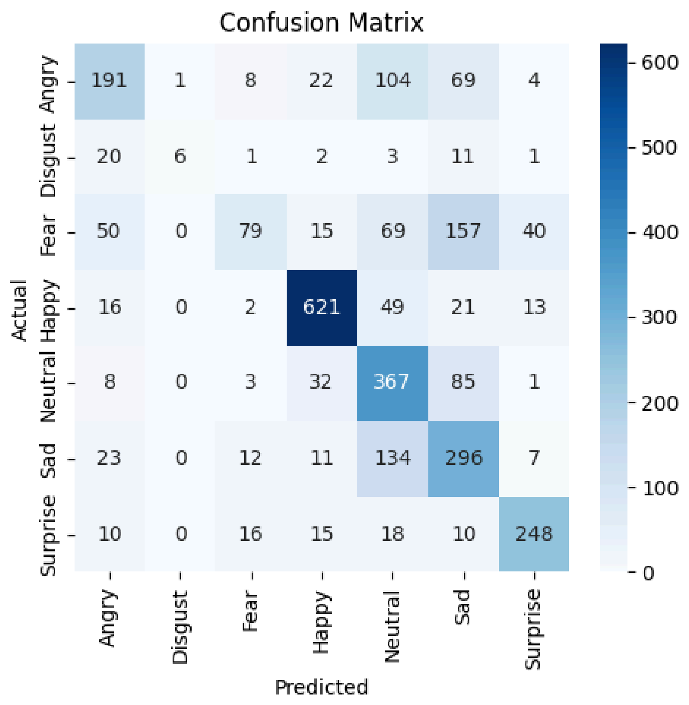
After training the Convolutional Neural Network (CNN) on the FER2013 dataset, the model achieved a final validation accuracy of 62.97%. This is a reasonable result given the challenges of emotion classification, especially when using grayscale images that are only 48x48 pixels in size. The dataset also includes a variety of facial angles, lighting conditions, and noise, which makes the task more complex.

The model was able to learn the general features of different emotions, such as smiles, frowns, or widened eyes, and apply this knowledge when analyzing unseen images. The confusion matrix, as seen below, shows higher accuracy for emotions like Happy or Neutral, which are easier to distinguish, and more confusion between similar expressions such as Surprise, Sad, and Angry. Disgust barely had any true positives due to the number of images available to be used for training for this specific emotion being very few in comparison to the other emotions.

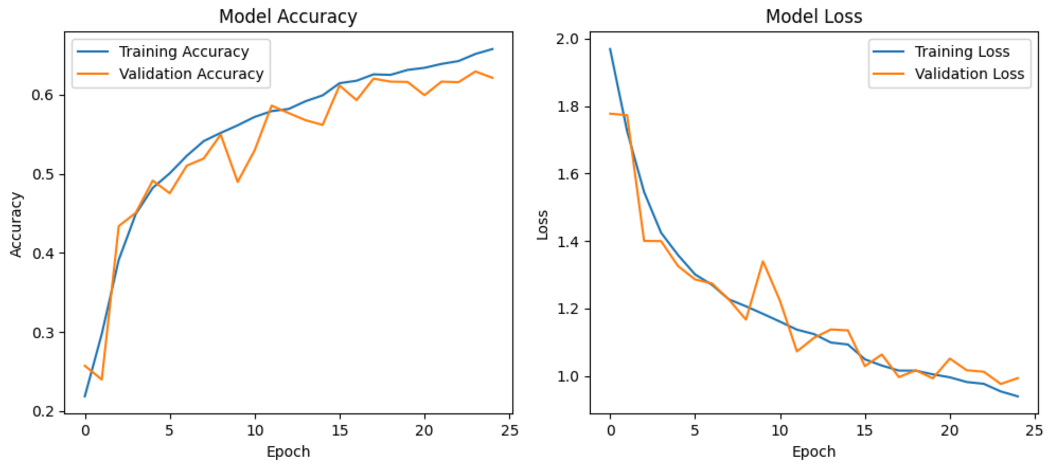


```
Test Accuracy: 0.6297
Test Loss: 0.9809
90/90 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step

Classification Report:
              precision    recall  f1-score   support

       Angry       0.60      0.48      0.53       399
     Disgust       0.86      0.14      0.24        44
        Fear       0.65      0.19      0.30       410
       Happy       0.86      0.86      0.86       722
     Neutral       0.49      0.74      0.59       496
         Sad       0.46      0.61      0.52       483
    Surprise       0.79      0.78      0.79       317

    accuracy                           0.63      2871
   macro avg       0.67      0.54      0.55      2871
weighted avg       0.66      0.63      0.61      2871
```

Classification Report for the Emotion Classification CNN



Confusion Matrix for the Emotion Classification CNN



Model Accuracy and Model Loss Plotted

Testing on Real Images

We tested the system on user-uploaded images containing one or more human faces. The MTCNN algorithm successfully detected most faces, even in cases where the faces were tilted, partially occluded, or varied in size. Each detected face was then processed by the CNN model to classify its emotion. In our tests, the pipeline worked smoothly. Faces were correctly detected, and emotions were classified with acceptable results. For example:

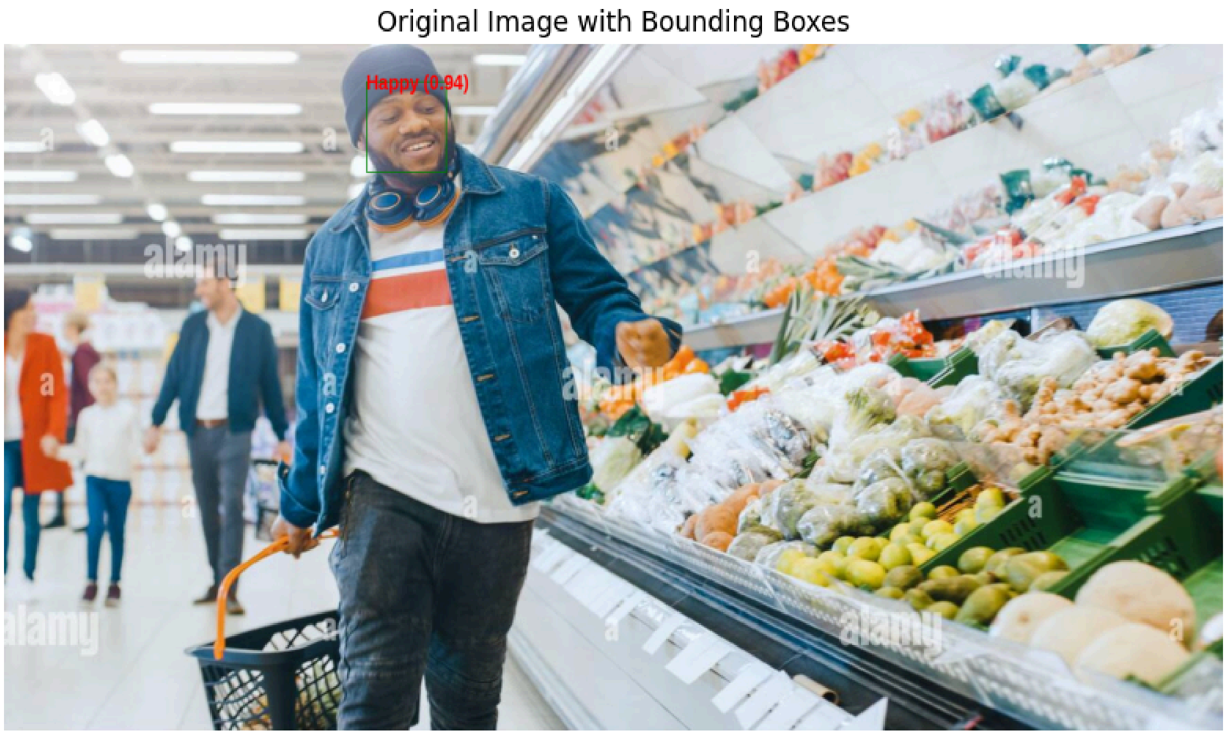


Image of a man in a supermarket after being processed through our program

We can see that in the image above, the face was detected successfully, classifying the customer as being "Happy", which we can see he is, as he is smiling.

In some cases, subtle emotions like Fear or Disgust were harder to classify accurately, which may be due to limited training data or overlap between similar facial features. The strengths of this program include the model's ability to work on real-world images, not just cropped face photos. The MTCNN handles multiple faces and varying face orientations well, and the CNN is fast and lightweight enough for real-time emotion classification. However, it has its drawbacks as well. The accuracy of the emotion classifier (62.97%) is not high enough for critical applications (e.g., security or health). Also, the model struggles with subtle or similar expressions (e.g., Fear vs. Disgust), and no context or body language is included — only facial expressions are analyzed.

As for potential improvements, using a higher quality or more diverse dataset (e.g., AffectNet, RAF-DB) can help increase model accuracy. Adding real-time analysis to capture changes in expression over time can also give more insights into customer satisfaction levels, as well as adding the ability to analyze body language, though it may be harder to implement as it is not necessarily as objective as emotion classification.

Deployment

Using a simple Python script, we managed to deploy the project that detects and classifies faces and emotions in real-time. The project's functionality can also be demonstrated through our code in the Colab Jupyter Notebook environment by uploading images. The final section of the notebook allows users to upload any image from their local device. Once uploaded, the following steps occur:

- The original image is first displayed.
- MTCNN is used to detect all visible faces in the image.
- Each face is cropped, resized, and preprocessed to match the CNN input requirements.
- The preprocessed face images are classified by the trained CNN model into one of the seven emotion categories: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise. Once each face has been classified into an emotion, the processed face image, along with its emotion, is displayed.
- Finally, the original image is displayed with bounding boxes and emotion labels over each detected face.

This notebook-based demonstration shows how the complete system works on real, unstructured data, just like it would in a deployed application. It confirms that the face detection and emotion classification pipeline functions correctly and provides useful feedback based on user input.

The Python script shows the project has room for improvement, as the video feed is slightly choppy, but it shows that it also has potential to develop further for real-world use cases. The demonstration of the deployment can be viewed in the link below:

https://youtu.be/oHWkWC97bl0

Conclusion

This project developed an emotion recognition system that identifies faces by using deep learning methods. A CNN was applied to the FER2013 dataset in order to arrange facial expressions into seven different emotions. First, MTCNN was used to detect the faces in the pictures and extract them, and then the detected faces were sent to the CNN to determine their emotions.

Since the dataset was only grayscale images at a low resolution, the 62.97% accuracy proves to be a reasonable outcome. It became apparent that Happy and Neutral emotions were easier for the model, but Fear and Disgust were more difficult because they look alike.

The solution was presented in a Python script for real-time detection and classification as well as a Colab Jupyter Notebook, allowing people to load an image and get predictions of emotion for all the faces that are identified. This way of working points to the usefulness of combining CNNs and MTCNN for real emotion analysis.

Even though it is not applicable to the real world now, this system may be helpful for companies that need to learn about customer emotions from their faces. Ongoing research could concentrate on enhancing accuracy by obtaining better data, making the program able to use knowledge from other fields, and designing an easy-to-use interface.

Team Member	Work Done	Contribution
MUHAMMAD AMIRUL HAZIQ BIN MUHAMAD HASMAHADI	Code Review, Report Writing	100%
MUHAMMAD UMAIR ASYRAAF BIN SAMSURI	Data preprocessing and preparation, Report Writing	100%
YUSUF MOHAMMAD YUNUS	Enhancing the model and adding the Face Detection model, Deploying the project	100%
SUHAIB ADNAAN	Creating, training, and testing the model, and adding the Face Detection model, Deploying the project	100%
ALNATSHEH HUTHIFA M J	Code Review, Report Writing	100%

References

FER-2013. (2020, July 19). Kaggle. Retrieved from, https://www.kaggle.com/datasets/msambare/fer2013

GeeksforGeeks. (2024a, June 6). *What is Face Detection?* GeeksforGeeks. https://www.geeksforgeeks.org/what-is-face-detection/

GeeksforGeeks. (2024, September 13). *A complete guide to face detection and face recognition in 2024*. GeeksforGeeks. https://www.geeksforgeeks.org/a-complete-guide-to-face-detection-and-face-recognition-in-2024/

GeeksforGeeks. (2025, April 3). *Introduction to Convolution Neural Network*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-convolution-neural-network/

GeeksforGeeks. (2025, May 29). *Convolutional Neural Network (CNN) in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1512.03385

Jawabreh, A. (2023, September 17). Multi-task Cascaded Convolutional Neural Network (MTCNN) | The Modern Scientist. *Medium*. https://medium.com/the-modern-scientist/multi-task-cascaded-convolutional-neural-network-mtcnn-a31d88f501c8

Lxyuan. (2019, July 21). *Facial Expression Recognition using CNN*. Kaggle. https://www.kaggle.com/code/lxyuan0420/facial-expression-recognition-using-cnn/notebook

Mollahosseini, A., Chan, D., & Mahoor, M. H. (2016). Going deeper in facial expression recognition using deep neural networks. *IEEE Workshop on Applications of Computer Vision (WACV)*, 1–10. https://doi.org/10.1109/wacv.2016.7477450

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, *23*(10), 1499–1503. https://doi.org/10.1109/lsp.2016.2603342