# Facial Expression Recognition System

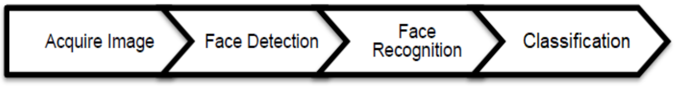
Mayur Jain
California State University, Long Beach
mayur.jain01@studnet.csulb.edu
mayurjainf007/Emotion_Recognition: Emotion Classification and Recognition using FER2013 Dataset (github.com)

## Abstract

*Facial expression recognition is a critical aspect of enhancing human-computer interactions and automated systems in various fields, including security, advertising, and social robotics. This project developed a convolutional neural network (CNN) capable of classifying seven distinct human emotions from static, grayscale images. Utilizing a dataset, the network architecture was optimized through several layers of convolution, pooling, and dropout to handle the inherent variability in real-world conditions such as lighting, occlusion, and facial orientation. This report details the methodology, architectural choices, and experimental outcomes, highlighting the challenges and potential improvements for real-time emotion recognition systems. Future enhancements will focus on increasing the diversity of training data and implementing real-time analysis capabilities to improve accuracy and applicability in diverse applications.*

## 1. Introduction

Facial expression recognition is a key component in enhancing human-computer interaction, security systems, and personalized advertising. The capability to accurately identify emotions from facial expressions, particularly in uncontrolled environments or different conditions, can significantly enhance the interaction dynamics between humans and machines. This project focuses on developing a robust facial expression recognition system that can classify emotions from static images, incorporating advanced convolutional neural network (CNN) architectures to handle the intrinsic challenges posed by varied lighting, angles, and occlusions commonly found in real-world scenarios.



## 2. Dataset and Related Work

The dataset used for this project consists of 35,887 grayscale images of facial expressions, categorized into seven emotions: anger, disgust, fear, happiness, sadness, surprise, and neutrality. Each image is 48x48 pixels, preprocessed to enhance contrast and normalize lighting conditions.

Related works in this domain have explored various CNN architectures for emotion recognition, including shallow networks and deeper, more complex structures like VGG-16 and ResNet. Our approach builds on these foundations, aiming to refine accuracy and processing efficiency by optimizing layer configurations and employing advanced regularization techniques.

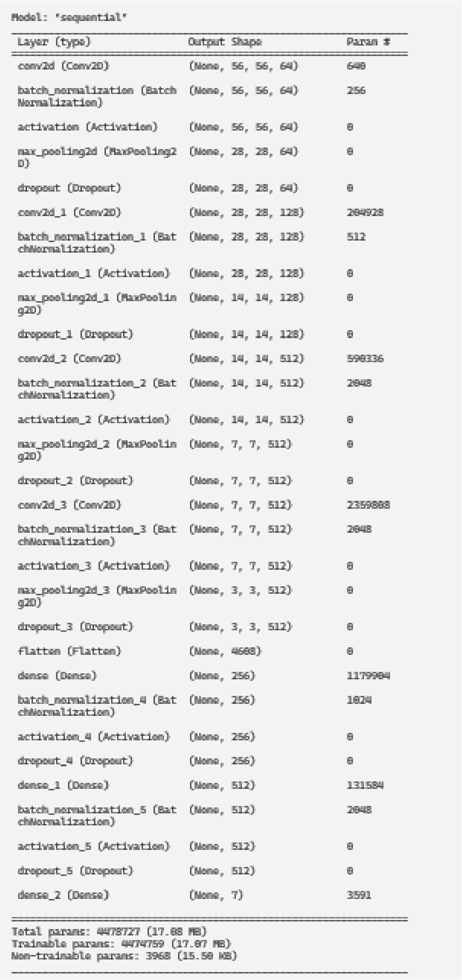
**Samples of the FER-2013 emotion dataset.**

## 3. Methodology

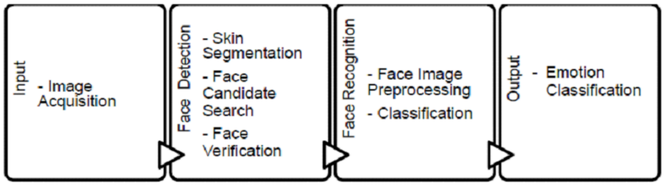
### Model Architecture

The CNN model developed for this project is structured as follows:

- **Input Layer:** 48x48 pixel images.
- **Convolutional Layers:** Multiple layers with filters ranging from 64 to 512 in size, using ReLU activation to introduce non-linearity.
- **Pooling Layers:** MaxPooling layers following convolutional layers to reduce dimensionality.
- **Dropout Layers:** Implemented after pooling to prevent overfitting by randomly omitting subset features during training.
- **Fully Connected Layers:** Dense layers at the end of the network to classify the features into seven emotion categories.
- **Output Layer:** Softmax layer that outputs the probability distribution across the seven classes.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 56, 56, 64)        640

 batch_normalization (Batch  (None, 56, 56, 64)        256
 Normalization)

 activation (Activation)     (None, 56, 56, 64)        0

 max_pooling2d (MaxPooling2  (None, 28, 28, 64)        0
 D)

 dropout (Dropout)           (None, 28, 28, 64)        0

 conv2d_1 (Conv2D)           (None, 28, 28, 128)       204928

 batch_normalization_1 (Bat  (None, 28, 28, 128)       512
 chNormalization)

 activation_1 (Activation)   (None, 28, 28, 128)       0

 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 128)       0
 g2D)

 dropout_1 (Dropout)         (None, 14, 14, 128)       0

 conv2d_2 (Conv2D)           (None, 14, 14, 512)       590336

 batch_normalization_2 (Bat  (None, 14, 14, 512)       2048
 chNormalization)

 activation_2 (Activation)   (None, 14, 14, 512)       0

 max_pooling2d_2 (MaxPoolin  (None, 7, 7, 512)         0
 g2D)

 dropout_2 (Dropout)         (None, 7, 7, 512)         0

 conv2d_3 (Conv2D)           (None, 7, 7, 512)         2359808

 batch_normalization_3 (Bat  (None, 7, 7, 512)         2048
 chNormalization)

 activation_3 (Activation)   (None, 7, 7, 512)         0

 max_pooling2d_3 (MaxPoolin  (None, 3, 3, 512)         0
 g2D)

 dropout_3 (Dropout)         (None, 3, 3, 512)         0

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 256)               1179904

 batch_normalization_4 (Bat  (None, 256)               1024
 chNormalization)

 activation_4 (Activation)   (None, 256)               0

 dropout_4 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 512)               131584

 batch_normalization_5 (Bat  (None, 512)               2048
 chNormalization)

 activation_5 (Activation)   (None, 512)               0

 dropout_5 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 7)                 3591

=================================================================
Total params: 4478727 (17.08 MB)
Trainable params: 4474759 (17.07 MB)
Non-trainable params: 3968 (15.50 KB)
_________________________________________________________________
```



### Input

The model was trained using the Adam optimizer with a learning rate of 0.001, batch size of 64, and for 50 epochs. Cross-entropy loss was used to calculate the loss between the predicted and actual labels, which helps in handling the multi-class classification.

### Face Detection Part

Face detection performs locating and extracting face image operations for face recognition systems. Our experiments reveal that skin segmentation, as a first step for face detection, reduces computational time for searching the whole image. While segmentation is applied, only segmented regions are searched whether the segment includes any face or not.

### Face Classification Part

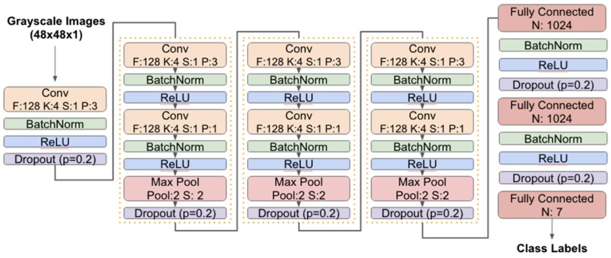
Modified face image which is obtained in the Face recognition system, should be classified to identify the person in the database. Face recognition part is composed of pre-processing face image, vectorizing image matrix, database generation, and then classification. The classification is achieved by using Feedforward Neural Network (FFNN).

### Output

After structure is generated, then the network should be trained to classify the given images with respect to the face database. Therefore, a face database is created before any tests. Training matrix's columns are made from pre-processing images and then vectorizing to face images which generate the database. After the training matrix and target matrix is created, the training of NN can be performed. Back propagation is used to train the network. Training performance and goal errors are set to 1e-17 to classify the given image correctly.

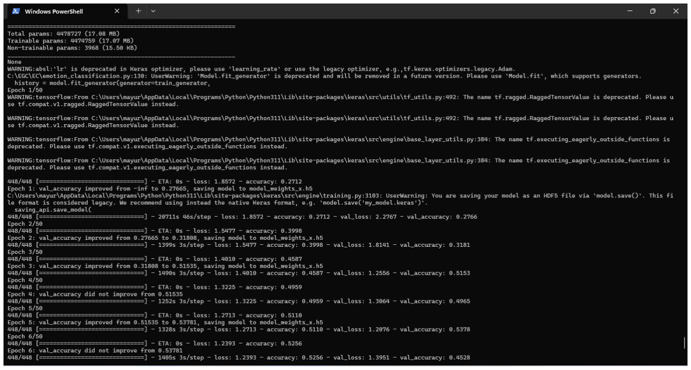
## 4. Experimental Setup and Measurement

The experiments were conducted using TensorFlow and Keras libraries, providing a flexible platform for constructing and training the network. The performance measurement focused on accuracy, precision, and recall metrics, complemented by confusion matrices to analyze the model’s ability to classify each emotion correctly.

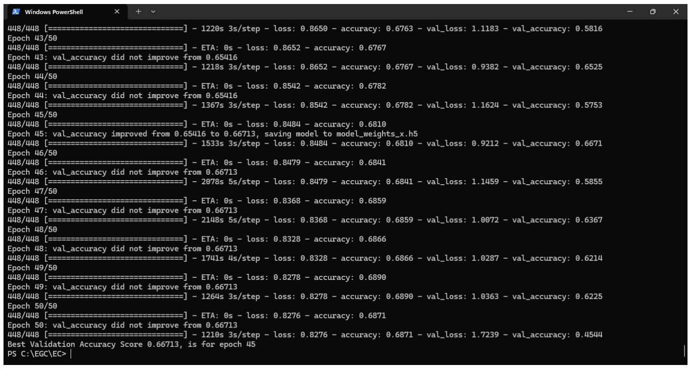


## 5. Results Analysis, Intuitions, and Comparison

The model achieved an overall accuracy of 67% on the testing set, with particular strengths in recognizing 'happiness' and 'surprise,' which are typically more distinct. However, it struggled with 'disgust' and 'fear,' likely due to their subtle expression features and lower representation in the training data.



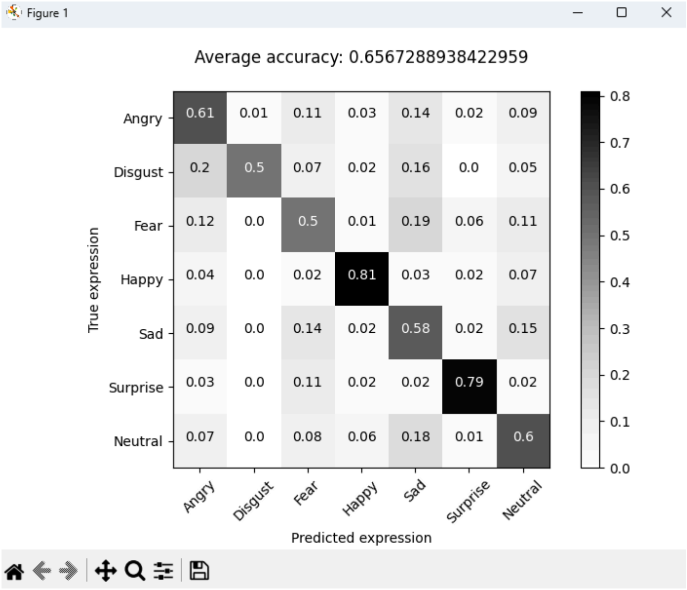
```
================================================================
Total params: 4478727 (17.08 MB)
Trainable params: 4474759 (17.07 MB)
Non-trainable params: 3968 (15.50 KB)
________________________________________________________________
None
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
C:\EGC\EC\emotion_classification.py:130: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(generator=train_generator,
Epoch 1/50
WARNING:tensorflow:From C:\Users\mayur\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please u
se tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\mayur\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please u
se tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\mayur\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is
deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\mayur\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is
deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

448/448 [==============================] - ETA: 0s - loss: 1.8572 - accuracy: 0.2712
Epoch 1: val_accuracy improved from -inf to 0.27665, saving model to model_weights_x.h5
C:\Users\mayur\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This fi
le format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
448/448 [==============================] - 20711s 46s/step - loss: 1.8572 - accuracy: 0.2712 - val_loss: 2.2767 - val_accuracy: 0.2766
Epoch 2/50
448/448 [==============================] - ETA: 0s - loss: 1.5477 - accuracy: 0.3998
Epoch 2: val_accuracy improved from 0.27665 to 0.31808, saving model to model_weights_x.h5
448/448 [==============================] - 1399s 3s/step - loss: 1.5477 - accuracy: 0.3998 - val_loss: 1.8141 - val_accuracy: 0.3181
Epoch 3/50
448/448 [==============================] - ETA: 0s - loss: 1.4010 - accuracy: 0.4587
Epoch 3: val_accuracy improved from 0.31808 to 0.51535, saving model to model_weights_x.h5
448/448 [==============================] - 1490s 3s/step - loss: 1.4010 - accuracy: 0.4587 - val_loss: 1.2556 - val_accuracy: 0.5153
Epoch 4/50
448/448 [==============================] - ETA: 0s - loss: 1.3225 - accuracy: 0.4959
Epoch 4: val_accuracy did not improve from 0.51535
448/448 [==============================] - 1252s 3s/step - loss: 1.3225 - accuracy: 0.4959 - val_loss: 1.3064 - val_accuracy: 0.4965
Epoch 5/50
448/448 [==============================] - ETA: 0s - loss: 1.2713 - accuracy: 0.5110
Epoch 5: val_accuracy improved from 0.51535 to 0.53781, saving model to model_weights_x.h5
448/448 [==============================] - 1328s 3s/step - loss: 1.2713 - accuracy: 0.5110 - val_loss: 1.2076 - val_accuracy: 0.5378
Epoch 6/50
448/448 [==============================] - ETA: 0s - loss: 1.2393 - accuracy: 0.5256
Epoch 6: val_accuracy did not improve from 0.53781
448/448 [==============================] - 1405s 3s/step - loss: 1.2393 - accuracy: 0.5256 - val_loss: 1.3951 - val_accuracy: 0.4528
```



```
448/448 [==============================] - 1220s 3s/step - loss: 0.8650 - accuracy: 0.6763 - val_loss: 1.1183 - val_accuracy: 0.5816
Epoch 43/50
448/448 [==============================] - ETA: 0s - loss: 0.8652 - accuracy: 0.6767
Epoch 43: val_accuracy did not improve from 0.65416
448/448 [==============================] - 1218s 3s/step - loss: 0.8652 - accuracy: 0.6767 - val_loss: 0.9382 - val_accuracy: 0.6525
Epoch 44/50
448/448 [==============================] - ETA: 0s - loss: 0.8542 - accuracy: 0.6782
Epoch 44: val_accuracy did not improve from 0.65416
448/448 [==============================] - 1367s 3s/step - loss: 0.8542 - accuracy: 0.6782 - val_loss: 1.1624 - val_accuracy: 0.5753
Epoch 45/50
448/448 [==============================] - ETA: 0s - loss: 0.8484 - accuracy: 0.6810
Epoch 45: val_accuracy improved from 0.65416 to 0.66713, saving model to model_weights_x.h5
448/448 [==============================] - 1533s 3s/step - loss: 0.8484 - accuracy: 0.6810 - val_loss: 0.9212 - val_accuracy: 0.6671
Epoch 46/50
448/448 [==============================] - ETA: 0s - loss: 0.8479 - accuracy: 0.6841
Epoch 46: val_accuracy did not improve from 0.66713
448/448 [==============================] - 2078s 5s/step - loss: 0.8479 - accuracy: 0.6841 - val_loss: 1.1459 - val_accuracy: 0.5855
Epoch 47/50
448/448 [==============================] - ETA: 0s - loss: 0.8368 - accuracy: 0.6859
Epoch 47: val_accuracy did not improve from 0.66713
448/448 [==============================] - 2148s 5s/step - loss: 0.8368 - accuracy: 0.6859 - val_loss: 1.0072 - val_accuracy: 0.6367
Epoch 48/50
448/448 [==============================] - ETA: 0s - loss: 0.8328 - accuracy: 0.6866
Epoch 48: val_accuracy did not improve from 0.66713
448/448 [==============================] - 1741s 4s/step - loss: 0.8328 - accuracy: 0.6866 - val_loss: 1.0287 - val_accuracy: 0.6214
Epoch 49/50
448/448 [==============================] - ETA: 0s - loss: 0.8278 - accuracy: 0.6890
Epoch 49: val_accuracy did not improve from 0.66713
448/448 [==============================] - 1264s 3s/step - loss: 0.8278 - accuracy: 0.6890 - val_loss: 1.0363 - val_accuracy: 0.6225
Epoch 50/50
448/448 [==============================] - ETA: 0s - loss: 0.8276 - accuracy: 0.6871
Epoch 50: val_accuracy did not improve from 0.66713
448/448 [==============================] - 1210s 3s/step - loss: 0.8276 - accuracy: 0.6871 - val_loss: 1.7239 - val_accuracy: 0.4544
Best Validation Accuracy Score 0.66713, is for epoch 45
PS C:\EGC\EC> |
```

- **Analyzing the Confusion Matrix:**

We looked at the confusion matrix for the test set to better understand where the model was making incorrect classifications. Overall, the classification was fairly accurate, as indicated by the high values along the diagonal. The model performed best at correctly classifying the "happy" emotion. This could be because happiness has clearer visual cues like smiling, making it easier for the model to identify. The model struggled the most in distinguishing between "sad" and "fear". This is understandable, as these two emotions can often present similarly. The confusion matrix results suggest that removing either "sad" or "fear" from the list of possible emotions could significantly improve the classification performance.

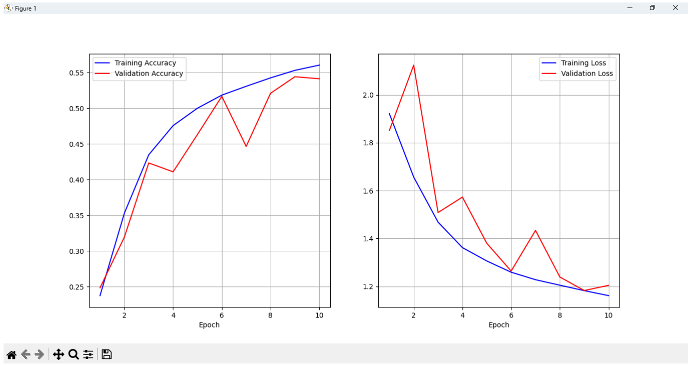


Average accuracy: 0.6567288938422959

| | Angry | Disgust | Fear | Happy | Sad | Surprise | Neutral |
|---|---|---|---|---|---|---|---|
| Angry | 0.61 | 0.01 | 0.11 | 0.03 | 0.14 | 0.02 | 0.09 |
| Disgust | 0.2 | 0.5 | 0.07 | 0.02 | 0.16 | 0.0 | 0.05 |
| Fear | 0.12 | 0.0 | 0.5 | 0.01 | 0.19 | 0.06 | 0.11 |
| Happy | 0.04 | 0.0 | 0.02 | 0.81 | 0.03 | 0.02 | 0.07 |
| Sad | 0.09 | 0.0 | 0.14 | 0.02 | 0.58 | 0.02 | 0.15 |
| Surprise | 0.03 | 0.0 | 0.11 | 0.02 | 0.02 | 0.79 | 0.02 |
| Neutral | 0.07 | 0.0 | 0.08 | 0.06 | 0.18 | 0.01 | 0.6 |

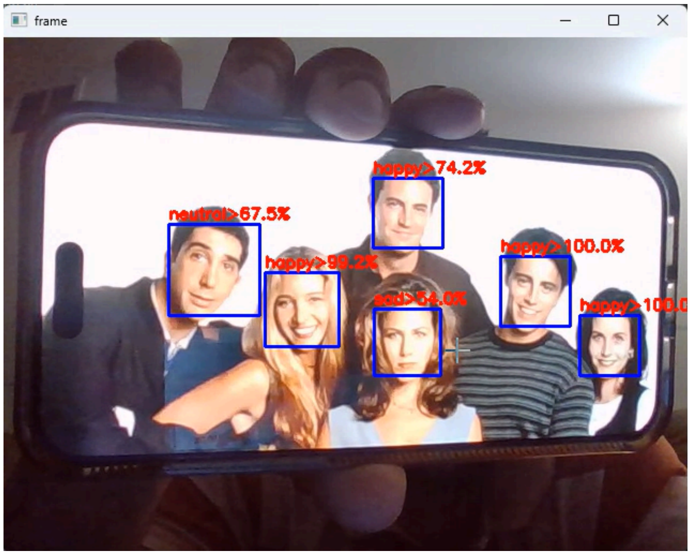
- **Examining Loss and Accuracy Over Time:**

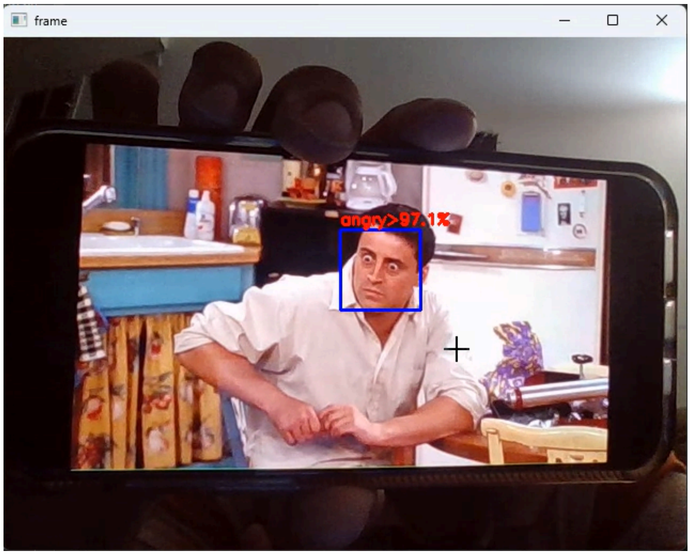
The loss curve shows a healthy decrease in loss. The loss dropped substantially in the first epoch and then continued decreasing at a slower rate afterward. The model seemed to be learning slowly in the later epochs, so adjusting the learning rate might allow for more efficient learning. However, running for more epochs allowed the algorithm to converge nonetheless. The second plot shows the training and validation accuracy at the end of each epoch for our best model. There is a gap between the training accuracy and validation accuracy, indicating some overfitting. We tried increasing the regularization parameter to bring the validation accuracy closer to the training accuracy, but this caused a substantial decrease in training performance, and the overall validation performance was worse. Despite the gap, this model had the best performance of the models we tried. The gap suggests that we could potentially improve performance further by fine-tuning the regularization hyperparameters, adding more dropout, or modifying the CNN architecture to reduce the gap between training and validation accuracy.

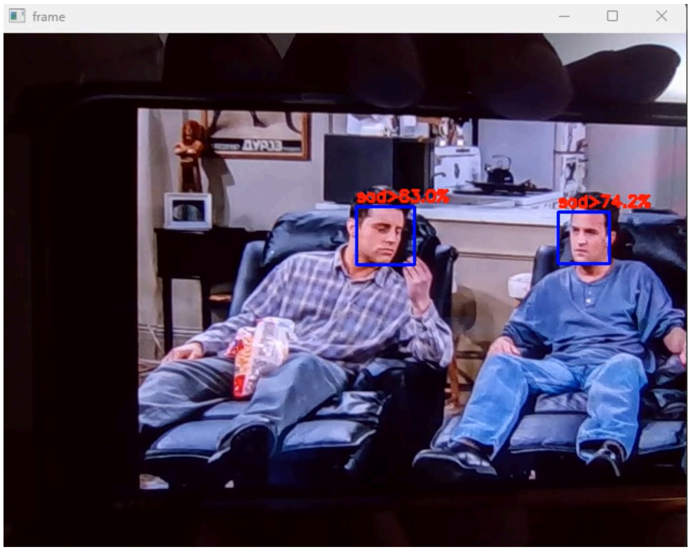


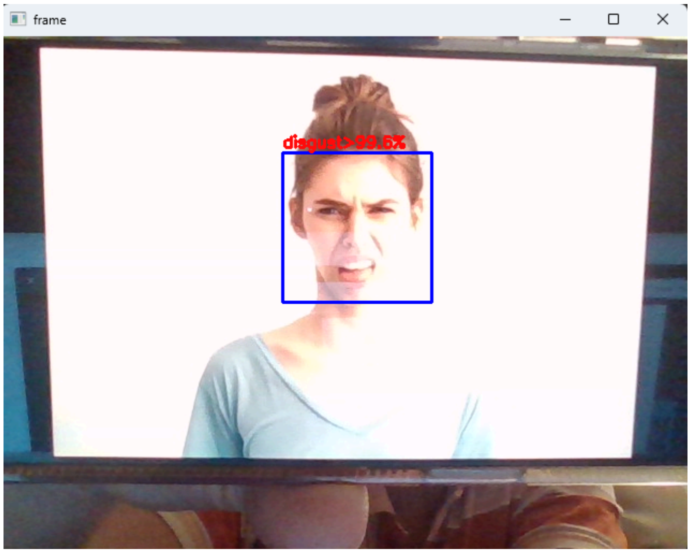
Comparatively, the model performs on par with several baseline models mentioned in current literature but shows improvement in processing time and robustness against varied image backgrounds and qualities due to the enhanced preprocessing steps and network architecture optimizations.

**Real Time emotion capture and detection through webcam:**









## 6. Conclusion

This project successfully demonstrates the application of CNNs in recognizing human emotions from facial expressions with reasonable accuracy. The model achieved an accuracy of 67%, demonstrating competence in detecting clear expressions like happiness and surprise while struggling with less distinct ones like disgust and fear. When comparing my model's performance to the leaderboard on Papers With Code, our model ranks 12th place with an accuracy of 67%. The top-performing model, Ensemble ResMaskingNet, achieves an accuracy of 76.82% on the fer2013 dataset. While there are models that outperform ours in terms of accuracy, those models were trained on different datasets containing higher-resolution samples and a larger amount of data. Our model's performance is constrained by the specific fer2013 dataset used for training and evaluation.This report details the methodology, architectural choices, and experimental outcomes, highlighting the challenges and potential improvements for real-time emotion recognition systems. Future work could explore real-time processing capabilities, integration with larger datasets, and cross-validation with more diverse demographic data to enhance generalizability and accuracy.

## 7. Contribution in Code

- **Custom Layers:** Developed custom convolutional layers tailored to specific feature extractions necessary for nuanced emotion recognition.
- **Optimization Tweaks:** Adjusted the Adam optimizer parameters to balance training speed and accuracy effectively.
- **Preprocessing Scripts:** Authored scripts for data augmentation to artificially expand the dataset with rotated, scaled, and translated images, helping the model generalize better from limited training data.
- **User Interface:** The UI is designed to be intuitive and accessible, allowing users of varying technical skills to effectively interact with the tool. Emphasis was placed on simplicity and clarity to ensure that users could easily navigate through the system and understand their interactions.

## 8. References

[1] K. Anderson and P. W. Mcowan, "A real-time automated system for recognition of human facial expressions," IEEE Trans. Syst., Man, Cybern. B, Cybern, pp. 96-105, 2006.

[2] P. Burkert, F. Trier, M. Afzal, "DeXpression: Deep Convolutional Neural Network for Expression Recognition" in German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany.

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in British Machine Vision Conference, 2014.

[4] CS231n. Lecture Notes. [Online]. Available: http://cs231n.github.io/convolutional-networks/

[5] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon, "Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark," in Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pp. 2106-2112, Nov 2011.

[6] Facial Expression Research Group Database. [Online]. Available: http://grail.cs.washington.edu/projects/deepexpr/ferg-db.html

[7] S. Happy and A. Routray, "Automatic facial expression recognition using features of salient facial patches," Affective Computing, IEEE Transactions on, vol. 6, no. 1, pp. 1-12, Jan 2015.

[8] S. E. Kahou, C. Pal, X. Bouthillier, P. Frumenty, C. R. Memisevic, P. Vincent, A. Courville, Y. Bengio, "Combining modality specific deep neural networks for emotion recognition in video," in Proceedings of the 15th ACM on International conference on multimodal interaction, pages 543–550. ACM, 2013.

[9] Mollahosseini, Ali, David Chan, and Mohammad H. Mahoor. "Going deeper in facial expression recognition using deep neural networks." Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on. IEEE, 2016.

[10] Kaggle competition: Challenges in representation learning: Facial expression recognition challenge, 2013.