Real-Time Face Emotion Recognition using Deep Convolutional Neural Networks

Fabian Meissner
University of Applied Sciences Ulm
Advanced Machine Learning
meisfa01@thu.de

***Abstract*—This paper presents a comprehensive approach to real-time face emotion recognition using Deep Convolutional Neural Networks (CNNs). We leverage a dataset of human face emotions to train and evaluate two CNN architectures: a custom 4-layer CNN and a deeper VGG-style CNN. The dataset comprises images across five emotion classes: Angry, Fear, Happy, Sad, and Surprise. We address class imbalance through data augmentation and class weighting. Hyperparameter optimization using Optuna is employed to fine-tune the model's performance. The final model achieves a validation accuracy of over 80% during training, significantly outperforming the initial trials. On the held-out test set of the original dataset, utilizing an 80/10/10 split, we achieved an accuracy of 91.17%. Evaluation on a custom self-recorded test set confirms this performance with an accuracy of 84%, highlighting both the model's robustness and specific weaknesses, particularly in recognizing ”Fear”.**

I. Introduction

Face emotion recognition is a critical component in human-computer interaction, affective computing, and psychological analysis [2]. Automated systems capable of discerning human emotional states from facial expressions have wide-ranging applications, from personalized user experiences to mental health monitoring [2], [3].

Traditional approaches often rely on handcrafted features, which may lack robustness against variations in lighting, pose, and occlusion. Deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized this field by automatically learning hierarchical feature representations from raw image data [3].

This work focuses on developing a robust face emotion recognition system. We explore two CNN architectures, perform extensive data analysis and preprocessing, and employ automated hyperparameter optimization to maximize performance. Furthermore, we bridge the gap between training and deployment by implementing a real-time inference pipeline that detects faces and classifies emotions from a live webcam feed.

II. Dataset and Preprocessing

A. Dataset Description

We utilize the ”Human Face Emotions” dataset [1], which contains grayscale images of human faces categorized into five emotion classes: Angry, Fear, Happy, Sad, and Surprise. The dataset is split into training, validation, and test sets to ensure robust evaluation.

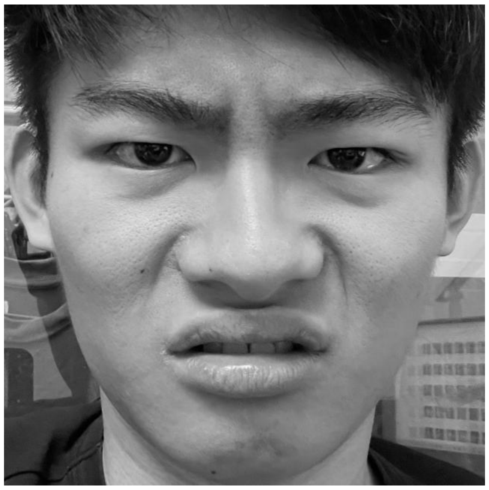


Fig. 1. Example image from the ”Angry” class in the training dataset.

B. Class Distribution and Balancing

An initial analysis of the dataset reveals a significant class imbalance:

- Happy: 18,439 samples
- Sad: 12,553 samples
- Angry: 10,148 samples
- Fear: 9,732 samples
- Surprise: 8,227 samples

”Happy” is the majority class, while ”Surprise” is the minority. Training on imbalanced data can lead to a model that is biased towards the majority class, achieving high accuracy by simply predicting the most frequent label while failing to recognize minority classes. To mitigate this, we employ class weighting in the Cross-Entropy Loss function. Weights are calculated inversely proportional to class frequencies, penalizing misclassifications of underrepresented classes more heavily.

C. Preprocessing and Augmentation

All images are resized to a fixed dimension of 64×64 pixels and converted to grayscale tensors. To improve model generalization and robustness, we apply data augmentation techniques to the training set, including:

- Random Horizontal Flip (p=0.5)
- Random Rotation (± 10 degrees)
- Random Affine transformations (translation up to 10%)

Validation and test sets undergo only resizing and normalization to ensure consistent evaluation metrics.

III. Methodology

A. Model Architectures

We investigate two CNN architectures:

1) EmotionCNN: A standard 4-layer CNN consisting of four convolutional blocks. Each block contains a Convolutional layer, ReLU activation, and Max Pooling. This is followed by a fully connected classifier head with Dropout for regularization.

2) DeepEmotionCNN: A deeper, VGG-style architecture designed to capture more complex features. The detailed architecture is presented in Table I.

TABLE I
DeepEmotionCNN Architecture

Block	Layers	Details
Block 1	2× Conv2d BatchNorm, ReLU MaxPool Dropout	32 filters, 3x3 kernel 2x2 $p = 0.25$
Block 2	2× Conv2d BatchNorm, ReLU MaxPool Dropout	64 filters, 3x3 kernel 2x2 $p = 0.25$
Block 3	2× Conv2d BatchNorm, ReLU MaxPool Dropout	128 filters, 3x3 kernel 2x2 $p = 0.25$
Classifier	Flatten Linear BatchNorm, ReLU Dropout Linear	 512 units $p \approx 0.2$ (Optimized) 5 outputs (Softmax)

B. Hyperparameter Optimization

We employ *Optuna* [4] to automate the search for optimal hyperparameters. The search space includes:

- **Batch Size**: $\{32, 64, 128\}$
- **Dropout Rate**: $[0.2, 0.6]$
- **Optimizer**: $\{$Adam, SGD, RMSprop$\}$
- **Learning Rate**: $[10^{-5}, 10^{-2}]$ (log scale)
- **Weight Decay**: $[10^{-6}, 10^{-3}]$ (log scale)

The optimization process utilizes the Tree-structured Parzen Estimator (TPE) sampler and Median Pruner to efficiently explore the hyperparameter space.

IV. Experiments and Results

A. Optimization Results

The hyperparameter optimization process evaluated 20 trials with 5 epochs each. Figure 3 illustrates the optimization history for the DeepEmotionCNN, showing the validation accuracy across different trials.

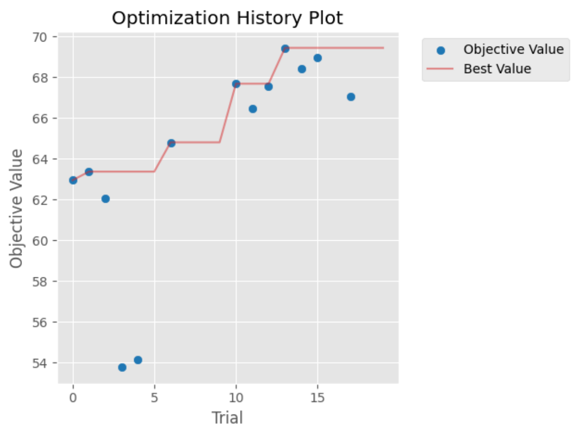


Fig. 3. Optuna optimization history for the DeepEmotionCNN showing the improvement in validation accuracy over trials.

The best trial achieved a validation accuracy of **69.44%** using the DeepEmotionCNN architecture. Key optimal parameters were:

- Batch Size: 64
- Dropout Rate: ≈ 0.20
- Optimizer: RMSprop
- Learning Rate: $\approx 6.41 \times 10^{-4}$
- Weight Decay: $\approx 1.45 \times 10^{-5}$

The more shallow EmotionCNN architecture achieved a significantly lower accuracy and is therefore not considered in the following chapters.

B. Training Dynamics

Following the hyperparameter optimization, where trials were limited to 5 epochs for computational efficiency,

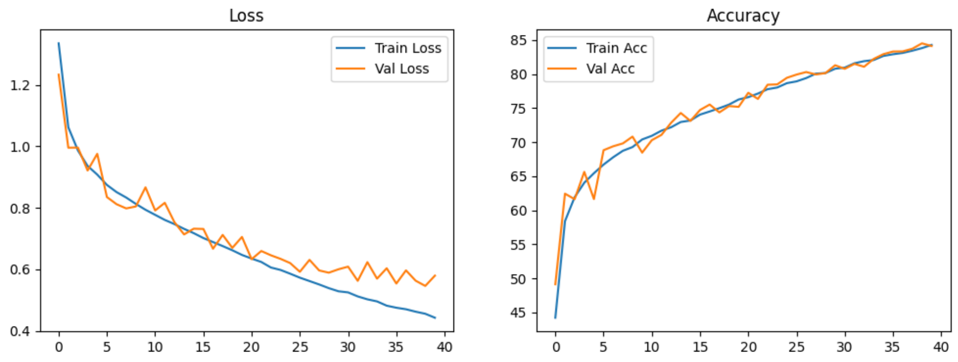


Fig. 2. Training and validation loss and accuracy for the final model. The validation accuracy reaches over 80%, outperforming the initial Optuna trials.

we trained the final model using the best parameters found. For the final training run, we extended the duration to 40 epochs to allow the model to fully converge. Figure 2 shows the training and validation loss and accuracy over these 40 epochs.

The training history shows a steady decrease in loss and improvement in accuracy. The validation accuracy tracks the training accuracy closely, indicating that the regularization techniques (Dropout, Data Augmentation) effectively prevented severe overfitting. Notably, the final model achieves a higher accuracy than the average Optuna trial, demonstrating the effectiveness of the selected hyperparameters.

Finally, we evaluated the model on the test partition of the original dataset. Utilizing an 80/10/10 split for training, validation, and testing, we achieved a test accuracy of 91.17% on this dataset. This further confirms the model's efficacy before evaluating it on the custom external dataset.

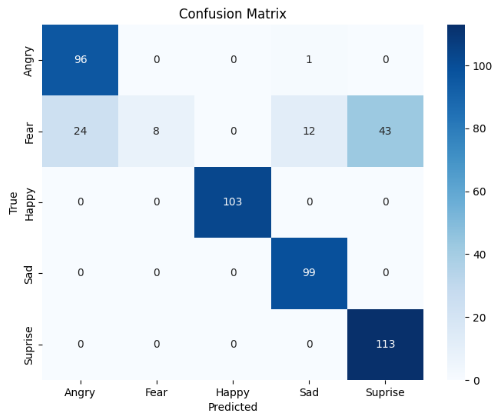
C. Evaluation on Custom Test Set

To verify the model's real-world performance, we collected a custom test set using the webcam inference pipeline. This dataset consists of 499 images across the 5 emotion classes, captured under natural lighting conditions.

The model achieved an overall accuracy of 84% on this custom dataset. The detailed classification report is shown below:

The results indicate excellent performance on ”Happy”, ”Sad”, and ”Surprise” classes. However, the model struggles significantly with the ”Fear” class, achieving a recall of only 0.09. Despite having perfect precision (no false positives for Fear), it fails to identify most actual Fear samples, likely misclassifying them as Surprise or Angry due to feature similarity (e.g., widened eyes). This suggests that the ”Fear” expression in the

Fig. 4. Example image from the ”Sad” class in the custom self-recorded test set.


Confusion Matrix
True
Predicted
Angry
Fear
Happy
Sad
Suprise

True \ Predicted	Angry	Fear	Happy	Sad	Suprise
Angry	96	0	0	1	0
Fear	24	8	0	12	43
Happy	0	0	103	0	0
Sad	0	0	0	99	0
Suprise	0	0	0	0	113

Fig. 5. Confusion matrix evaluated on the custom 499-image test set. The matrix highlights the model's strong performance on 'Happy', 'Sad', and 'Surprise' classes, while revealing significant misclassification for the 'Fear' class.

TABLE II
CLASSIFICATION REPORT ON CUSTOM TEST SET

Class	Precision	Recall	F1-Score	Support
Angry	0.80	0.99	0.88	97
Fear	1.00	0.09	0.17	87
Happy	1.00	1.00	1.00	103
Sad	0.88	1.00	0.94	99
Surprise	0.72	1.00	0.84	113
Overall Accuracy			**0.84**	**499**

training set might be distinct from the posed expressions in the test set, or that the model has learned to prioritize other classes despite the weighting.

D. Real-Time Inference

We developed a Python-based inference pipeline using OpenCV and PyTorch. The system: 1. Captures video frames from the webcam. 2. Detects faces using a Haar Cascade classifier. 3. Preprocesses the face region (crop, grayscale, resize). 4. Feeds the tensor to the trained DeepEmotionCNN model. 5. Applies a moving average filter (window size 10) to stabilize predictions. 6. Displays the emotion label and confidence score if it exceeds a threshold (0.5).

This pipeline successfully runs in real-time, providing immediate feedback on the user's emotional state.

V. Conclusion

This project demonstrated the end-to-end development of a face emotion recognition system. By moving from a simple CNN to a deeper architecture and employing rigorous hyperparameter optimization, we achieved a respectable accuracy of over 80% on the validation set and 84% on a custom test set.

While the model performs well on distinct emotions like Happiness and Sadness, the poor recall for Fear highlights the challenges of subtle emotion differentiation and dataset bias. Future work could address this by employing Focal Loss to penalize hard-to-classify examples more heavily. Additionally, collecting a dataset of spontaneous expressions, rather than posed ones, could improve real-world generalization. Transfer learning with larger pre-trained models (e.g., ResNet, VGG16) and more sophisticated face detection methods should also be explored to further improve performance and robustness across all classes.

Code Availability

The complete source code, including data preparation, training notebooks, and the inference script, is available in the project repository: https://github.com/meisfa01/emotion_recognition_paper

References

[1] Samith Sachidanandan, "Human Face Emotions Dataset," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/samithsachidanandan/human-face-emotions

[2] R. A. Calvo and S. K. D'Mello, "Affect Detection: An Interdisciplinary Review of Models, Methods, and Their Applications," *IEEE Transactions on Affective Computing*, vol. 1, no. 1, pp. 18–37, 2010.

[3] S. Li and W. Deng, "Deep Facial Expression Recognition: A Survey," *IEEE Transactions on Affective Computing*, vol. 13, no. 3, pp. 1195–1215, 2020.

[4] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2623–2631.