Revisiting Affective State Detection: A Comparative Analysis of Facial Expression Recognition Models in Human-Computer Interaction

Ali Ullah ALI.ULLAH@FAU.DE
Master of Science - Artificial Intelligence, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg
Matriculation Number: 23269239

Fida Hussain FIDA.HUSSAIN@FAU.DE
Master of Science - Artificial Intelligence, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg
Matriculation Number: 23209327

Hamza Naeem HAMZA.NAEEM@FAU.DE
Master of Science - Artificial Intelligence, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg
Matriculation Number: 23173252

Ghulam Mustafa GHULAM.MUSTAFA@FAU.DE
Master of Science - Artificial Intelligence, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg
Matriculation Number: 23186746

Submitted to: Prof. Dr.-Ing. Tobias Günther
Department of Computer Science
Friedrich-Alexander-Universität Erlangen
tobias.guenther@fau.de

Abstract

This report comprehensively studies affective state detection through facial expression analysis in human-computer interaction. Building upon the work of Samara et al., we replicate and extend their methodology using different computational models. Utilizing the "Corrective re-annotation of FER - CK+ - KDEF " (1) dataset, which encompasses a range of facial expressions, we apply advanced feature extraction techniques and facial landmark detection methods. Our study contrasts the performance of various models, including a Neural Network, Ensemble SVM, and standard SVM. It provides a comparative analysis of their efficacy in accurately detecting emotional states through facial expressions. The findings of this study contribute to the growing body of knowledge in the realm of affective computing and human-computer interaction, offering insights into the effectiveness of diverse computational approaches in emotion recognition.

1. Introduction

The field of human-computer interaction has increasingly recognized the importance of understanding and responding to user affective states. Affective state detection, mainly through facial expression analysis, is vital in developing more intuitive and responsive computer systems. This report delves into the realm of affective computing, aiming to reproduce and extend the work of Samara et al (2). The primary objective is to evaluate the efficacy of various computational models in recognizing and interpreting different facial expressions, thereby contributing to the development of more empathetic and user-aware computer interfaces. The importance of this study lies in its potential to enhance user experience in human-computer interactions and provide a deeper understanding of affective state detection instruments.

2. Related Work

Related work: The field of affective state detection through facial expressions represents a dynamic and evolving area of study. Comprehensive surveys like the one by Corneanu et al. (2016) have been instrumental in mapping the landscape of facial expression recognition methodologies, ranging from RGB and 3D approaches to advanced thermal and multimodal techniques (3). In parallel, the exploration of deep learning models, particularly Convolutional Neural Networks (CNNs), has significantly advanced the field. Lopes et al. (2017) provide insights into the application of CNNs in facial expression recognition, highlighting the challenges and solutions associated with limited data and training sample order(4). Furthermore, the integration of physiological signals in emotion analysis, as demonstrated by Koelstra et al. (2012) in the creation of the DEAP database, underscores the multidisciplinary nature of this research domain(5). This report builds upon these foundational studies, aiming to delve deeper into the comparative effectiveness of various computational models in the context of affective state detection.

3. Methodology

3.1 Dataset

3.1.1 Context

In the domain of facial expression recognition, machine learning models predominantly utilize industry-standard datasets such as FER, CK+, and KDEF. A detailed manual inspection of these datasets revealed a critical issue: a substantial number of images were incorrectly categorized, which could lead to inaccuracies in the training of models. "Corrective re-annotation of FER - CK+ - KDEF " dataset focuses on the corrective manual re-annotation of these images, aiming to enhance the dataset's accuracy and thereby improve the performance of facial expression recognition models.

3.1.2 Content

The revised dataset includes over 32,900 images, categorized into eight emotional states: anger, contempt, disgust, fear, happiness, neutrality, sadness, and surprise. The collection primarily consists of grayscale human faces, with some sketches, all in grayscale format. Each image is standardized to a size of 224 x 224 pixels, in PNG format. This uniformity in image quality and size ensures that the dataset is conducive to effective and precise processing by various machine learning algorithms.

The datasets used for this re-annotation process are:

- FER Dataset: Available from the Kaggle Facial Expression Recognition Challenge. (6)
- CK+ Dataset: Hosted on GitHub, courtesy of WuJie1010. (7)
- KDEF Dataset: Downloadable from the official KDEF portal.(8)

3.2 Facial Landmark Detection

In our study, we utilized Python programming language (9) with a combination of OpenCV (10) and dlib libraries (11) for facial landmark detection. OpenCV's Haar cascade classifier, loaded with "haarcascade_frontal_face_default.xml", was employed to detect faces in the images initially. This step was crucial as it provided the bounding box coordinates for each face. Subsequently, we applied dlib's shape predictor, loaded with the "shape_predictor_68_face_landmarks.dat", to accurately locate and predict 68 facial landmarks within the region of each detected face. We selected 49 points that were required for the implementation. This dual approach of using OpenCV for face detection and dlib for detailed facial landmark prediction ensured high accuracy and efficiency in identifying key facial features necessary for our analysis.

3.3 Feature Extraction

The feature extraction process involved meticulous steps in creating the CSV file "facial_landmarks_with_distances.csv", which encapsulates the facial landmark data. After detecting faces and predicting facial landmarks, we focused on extracting Euclidean distances between each pair of selected landmarks. We carefully filtered and selected specific landmarks to ensure relevance and accuracy. Utilizing 'itertools.combinations' (12), we efficiently computed distances between all possible pairs of the chosen landmarks. The total number of unique combinations when choosing 2 items at a time from a set of 49 unique items is 1,176. These distances were compiled into a well-structured CSV file. Each row in this file represented an image, including its category and the calculated distances between landmark pairs. This systematic process transformed raw facial landmark data into a comprehensive set of features, encapsulated in the CSV file, ready for further analysis and application in our study.

4. Models Used

To explore the efficacy of various computational models in affective state detection, we implemented three distinct approaches:

4.1 Neural Network

This model leverages deep learning algorithms to analyze facial features and their correlation with emotional states. The architecture and training parameters of the neural network are optimized for handling the complexity and subtlety of facial expressions.

We utilize Python's Pandas library (13) to load the dataset from a CSV file. The data consists of facial landmarks with distances, where we separate the features (X) and labels (y). The feature data is then normalized using StandardScaler from the sklearn library (14). This normalization is crucial to ensure that our model treats all features equally and isn't biased towards features with larger magnitudes.

4.1.1 One-Hot Encoding and Data Splitting

We employ OneHotEncoder (15) to transform categorical labels into a one-hot encoded format, a necessary step for multi-class classification with neural networks. The dataset is then split into training and test sets, with 20% of the data reserved for testing, ensuring a robust evaluation of the model's performance.

4.1.2 Neural Network Construction and Training

We construct a neural network using Keras (16), a high-level neural network API. The network comprises multiple dense layers with ReLU activation functions, interspersed with batch normalization layers. Batch normalization is used to improve the stability and performance of the neural network. Although sections of the code include dropout layers, they are commented out in the final model. The neural network's final layer uses a softmax activation function, suitable for multi-class classification.

The model is compiled with the Adam optimizer and categorical cross-entropy loss function, which are standard choices for classification tasks. We also implement early stopping to prevent overfitting. This is a technique where training is halted if the model's performance on the validation set does not improve for a specified number of epochs. The model is trained on the training set with a batch size of 32 and validated on the test set.

4.1.3 Model Evaluation

Post-training, we evaluate the model's performance using accuracy and F1 score, two common metrics for classification

4.2 Ensemble SVM

The ensemble SVM(17) approach combines multiple SVM classifiers to enhance the robustness and accuracy of emotion detection. This method aims to capitalize on the strengths

of individual SVMs while mitigating their limitations.

We start by loading our dataset, "facial_landmarks_with_distances.csv", which contains facial landmark distance features. The dataset is then divided into features (X) and labels (y), with 'Category' serving as the label. A standard train-test split is performed with 80% of the data used for training and 20% for testing, using a random seed for reproducibility.

4.2.1 Feature Standardization

We apply standardization(19) to our features, removing the mean and scaling to unit variance. This is a crucial step in SVM applications as it ensures that each feature contributes proportionally to the final decision function.

4.2.2 Model Development

The model development phase involves two key steps:

- Firstly, we define a base SVM model with a linear kernel and a regularization parameter C of 0.5. The linear kernel is chosen for its effectiveness in high-dimensional spaces, as is common with facial landmark data.
- Secondly, we employ an ensemble approach by integrating the base SVM model into a Bagging Classifier(18). This ensemble method combines the predictions of several base estimators to improve robustness and accuracy. We use 10 estimators in our ensemble with the same random seed for consistency.

4.2.3 Model Training and Evaluation

The ensemble SVM model is then trained on the preprocessed training data. Following training, we evaluate the model on the test set. The performance metric used is accuracy, which quantifies the proportion of correctly predicted instances.

4.2.4 Model Persistence

Finally, the trained model is saved to a file using joblib, ensuring that the model can be easily reloaded for future predictions or further analysis.

4.3 SVM

Support Vector Machine (SVM) is a powerful and versatile supervised machine learning algorithm, widely used for both classification and regression tasks. It operates by finding the hyperplane that best separates different classes in the feature space, optimizing for the largest margin between the closest points of the classes, known as support vectors. SVM is particularly effective in high-dimensional spaces and is robust against overfitting, especially in cases where the number of dimensions exceeds the number of samples.

We employed a Support Vector Machine (SVM) classifier. Using a standard 80-20 split ratio. The feature data was standardized using scikit-learn's StandardScaler, ensuring that the features had a mean of zero and unit variance, which is a common preprocessing step

for SVMs.

An SVM classifier with a linear kernel was instantiated, with the regularization parameter C set to 0.25. This classifier was trained on the training data, and its performance was evaluated on the test data.

Two key metrics, accuracy, and F1 score, were calculated to assess the classifier's effectiveness. The accuracy score provided a straightforward metric of correct classifications, while the F1 score offered a more nuanced view by considering both precision and recall, especially valuable in imbalanced datasets.

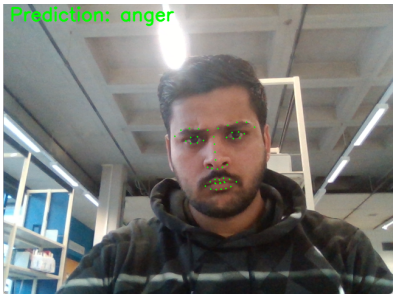
5. Results

Each of these models is evaluated based on their performance in accurately classifying emotional states from facial expressions, thus contributing to the broader understanding of affective state detection in human-computer interaction. Using the dataset 'Corrective re-annotation of FER - CK+ - KDEF' with all three models the following results are produced.

Table 1: Results

Model	**Configuration**	**Accuracy**	**F1 Score**
Neural Network	4 Layers densely connected neural net Input Dimension # 1176 Output Dimension #6 Optimizer 'Adams' (Learning Rate: 0.001) Epoch #30 Batch Size: 32	0.59	0.60
Ensemble SVM	Linear Kernel Regularization Parameter= 0.5 #SVM=10	0.62	0.60
SVM	Linear Kernel Regularization Parameter= 0.25	0.62	0.60

(a) Happiness



(b) Anger

Figure 1: Emotions Detected: Happiness and Anger

(a) Surprise

(b) Neutral

Figure 2: Emotions Detected: Surprise and Neutral

6. Discussion

6.1 Results and Future Directions

The results presented in Table 1 indicate a relatively low accuracy and F1 score for the implemented machine learning models, including a Neural Network, an Ensemble SVM, and a standard SVM. The accuracy levels across these models hover around 0.59 to 0.62, which suggests that the current approach, particularly the method of feature extraction using distances between facial landmarks, may not be optimally capturing the complexities of the data.

In light of these findings, our future work will pivot towards employing Convolutional Neural Networks (CNNs) for feature extraction. CNNs excel in extracting both low-level and high-level features directly from pixel data, thus potentially offering a more nuanced understanding of the facial images. This approach sidesteps the limitations of manually engineered features, such as landmark distances, by leveraging the inherent pattern recognition capabilities of CNNs. We anticipate that this shift to a more sophisticated feature extraction method will enhance the model's ability to accurately classify and interpret facial expressions, thereby improving both accuracy and F1 score metrics in our subsequent experiments.

6.2 Future Work

For the next milestone, we plan to focus on further developing and testing our model. The key areas of exploration will include:

- Exploring the application of different networks and approaches, such as integrating face detectors with pre-trained models, and advanced detection algorithms like Convolutional Neural Networks (CNN), and YOLO (You Only Look Once).
- Evaluating the trade-offs between model accuracy and computational efficiency to ensure the feasibility of real-time processing.
- Establishing benchmarks for model performance using suitable evaluation metrics, including accuracy, F1-score, and confusion matrices.
- Developing a robust training and validation pipeline, which will encompass techniques for cross-validation and hyperparameter tuning.
- Planning for the deployment of the model and its integration into a real-time system.

References

[1] Corrective Re-annotation of FER-CK-KDEF. *Sudarshan Vaidya (2020).* The dataset is available at: https://www.kaggle.com/datasets/sudarshanvaidya/corrective-reannotation-of-fer-ck-kdef

[2] "Affective State Detection via Facial Expression Analysis within a Human–Computer Interaction Context," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 2175–2184, 2019, doi: 10.1007/s12652-017-0636-8.

[3] Corneanu C.A., Simon M.O., Cohn J.F., Guerrero S.E. (2016) "Survey on RGB, 3D, Thermal, and Multimodal Approaches for Facial Expression Recognition: History, Trends, and Affect-Related Applications," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 8, pp 1548-1568.

[4] Lopes A.T., de Aguiar E., De Souza A.F., Oliveira-Santos T. (2017) "Facial Expression Recognition with Convolutional Neural Networks: Coping with Few Data and the Training Sample Order," Pattern Recognition, vol. 61, pp 610-628.

[5] Koelstra S., Muhl C., Soleymani M., Lee J-S., Yazdani A., Ebrahimi T., Pun T., Nijholt A., Patras I. (2012) "DEAP: A Database for Emotion Analysis; Using Physiological Signals," IEEE Transactions on Affective Computing, vol. 3, no. 1, pp 18-31.

[6] FER Dataset, *Challenges in Representation Learning: Facial Expression Recognition Challenge*, Available at: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

[7] CK+ Dataset, *Facial Expression Recognition on CK+ dataset*, Hosted by WuJie1010 on GitHub, Available at: https://github.com/WuJie1010/Facial-Expression-Recognition.Pytorch/tree/master/CK%2B48

[8] KDEF Dataset, *Karolinska Directed Emotional Faces*, Available at: https://www.kdef.se/download-2/register.html

[9] "Python Documentation." Python Software Foundation. Available at: https://docs.python.org/3/

[10] "OpenCV Documentation." OpenCV. Available at: https://docs.opencv.org/master/

[11] "DLIB Library." Davis E. King. Available at: http://dlib.net/

[12] "Itertools — Functions creating iterators for efficient looping." Python Software Foundation. Available at: https://docs.python.org/3/library/itertools.html

[13] McKinney, W. (2010). "Data Structures for Statistical Computing in Python." In *Proceedings of the 9th Python in Science Conference*, pp. 51-56. Available at: https://pandas.pydata.org/

[14] Pedregosa et al. (2011). "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, 12, pp. 2825-2830. Available at: `https://scikit-learn.org/`

[15] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). "Array programming with NumPy." *Nature*, 585(7825), 357-362. Available at: `https://numpy.org/`

[16] Chollet, F., et al. (2015). "Keras." Available at: `https://keras.io/`

[17] Cortes, C., & Vapnik, V. (1995). "Support-Vector Networks." *Machine Learning*, 20(3), 273-297.

[18] Breiman, L. (1996). "Bagging Predictors." *Machine Learning*, 24(2), 123-140.

[19] Jolliffe, I.T. (2002). "Principal Component Analysis and Factor Analysis." In *Principal Component Analysis*, Springer Series in Statistics, 115-128.