TU/e – Dept. of Electrical Engeneering
Data Analysis & Learning Methods

Real-time Emotion Recognition Using Computer Vision and Machine Learning Models

Group 42
Authors:
S. Chinniah (2176467)
P. Kuo (2354780)
G. Salvo (2140004)

During the preparation of this work, we used Google Gemini Flash 2.5 and ChatGPT 5 Extended Thinking to assist with the report writing process particularly for research assistance, problem understanding and for minor gramatical improvement recommendations. After using this tool/service, Shivun Chinnniah, Pochun Kuo and Gabriele Salvo evaluated the validity of the tool's outputs, including the sources that generative AI tools have used, and edited the content as needed. As a consequence, Shivun Chinnniah, Pochun Kuo and Gabriele Salvo take full responsibility for the content of their work.

Contents

1 Introduction

Real-time emotion recognition for automated systems offers a solution to bridge the gap between machines and humans. This report documents the development of a real-time emotion recognition system that can be deployed on personal computers with web cameras and potentially other camera-enabled mobile devices. A combination of Computer Vision techniques for face detection using the *OpenCV* [1] and *Dlib* [2] libraries as well as *Python*-based machine learning (ML) tool kits (*scikit-learn*[3]). Four classifier machine learning models will be implemented for the task of emotion recognition. These are: a *State Vector Machine* (SVM), a *Multi-layer Perceptron* (MLP), a *Random Forrest*, and a *Fuzzy Classifier*.

In 2013 the FER competition was held, in which competitors had the task of developing a facial recognition ML model using a common dataset. The data set from this challenge [4] is the primary source of training data for this project.

This report will first provide an exploration of the FER201 dataset, explain the general data pre-processing and feature extraction techniques used. An explanation of the training process of each model as well as their specific feature extraction requirements is provided. A performance evaluation of the models is then presented. Lastly, the application of the most suitable model in a real-time emotion detection demo is discussed.

2 Data Exploration

The FER2013 data set is a collection of 48×48 pixel grayscale images of various faces originally sourced from *Google image search* of various key words related to emotion. Subset of the *Google image*-sourced images was selected for the competition with the 35877 selected images belonging to one of the seven emotion label classes: "Anger", "Disgust", "Fear", "Happy", "Neutral", "Sad", and "Surprise". [5, 4].

According to [5] the human accuracy in classifying the FER-2013 data set was 65%, which is relatively low for such a common, and trivial task for humans. A visual inspection of a subset the data set images shows that the quality of the labeling is inconsistent. Some images clearly and unambiguously display the emotion depicted in the label, while for others it is rather ambiguous. Figure 1 shows 'good' (emotion corresponds with label) and 'bad' (emotion is ambiguous or label is incorrect) samples from the dataset to illustrate this. Hence a good accuracy metric would be in the order of 50-65%. According to [5] the best performing ML model achieved an accuracy score over 74%, however this could be due to overfitting. Care should be used when extrapolating the purpose of the dataset for facial-based emotion recognition.

Figure 1 also highlights some of the possible challenges that can be anticipated with classification. For instance, in some images: the subject (the human face) is not centered in the frame (ID: 30459); emotion is given by pose, or body language rather than by facial features alone (IDs: 16657, 18713, 28997); the subject is humanoid and not human (ID: 17294); the image quality is poor (IDs:29684 & 26795 are watermarked, IDs: 4564 & 992 have a different focus than the other images); And other inconsistencies related to image quality and labeling requiring more context to unambiguously identify emotion.

Figure 1: Good and bad samples from FER-2013 data set and class labels

Another influential characteristic of the data set that is essential to consider is the balance or distribution of the class labels. Table 1 shows the count and percentage share of the various labels for the *PrivateTest*, *PublicTest*, *Training*, and *Total* training-evaluation splits which are also predesignated as part of the FER-2013 data set. The *PrivateTest* was not shared to the public for the FER-2013 challenge, for this project it is used to evaluate the final performance of the models (it will be referred to as the "evaluation" data set), where as the *PublicTest* was shared to the public (it will be referred to as the "validation" data set, hence the distinction in name between *Private* and *Public* test data sets. Both evaluation and validation data sets each form 10% of the data samples, while the test data set is the remaining 80%. Table 1 highlights the extreme data imbalance between the classes.

As there are seven classes, an ideal distribution would

Table 1: Emotion Label (Class) distribution among *PrivateTest*, *PublicTest* and *Training* Data sets

	PrivateTest		PublicTest		Training		Total All Sets	
Emotion	Count	%	Count	%	Count	%	Count	%
Angry	491	13.68	467	13.01	3995	13.92	4953	13.80
Disgust	55	1.53	56	1.56	436	1.52	547	1.52
Fear	528	14.71	496	13.82	4097	14.27	5121	14.27
Happy	879	24.49	895	24.94	7215	25.13	8989	25.05
Sad	594	16.55	653	18.19	4830	16.82	6077	16.93
Surprise	416	11.59	415	11.56	3171	11.05	4002	11.15
Neutral	626	17.44	607	16.91	4965	17.29	6198	17.27
Total	3589	100	3589	100	28709	100	35887	100.00

have each class as 14.3% of the total data set. In this case, "Disgust" is only 1.5% of the data set, which is on the order of ten times smaller than the ideal. In contrast, the "Happy" class label is almost twice the size compared to the ideal share. The other class labels are thus also not at an ideal share, however their imbalance is less significant.

This large imbalance must be taken into account when developing the ML models as class labels which have more data will be more represented than other labels result in a model where the features or patterns of the more represented class are 'favored' in the learning processes, resulting in poor performance in the less represented class labels. Figure 2 shows the 'average' of 120 images from the training data set. It is evident that the associated emotion is discernible in these average "faces". Although this observation alone does not guarantee mathematical or statistical significance for a sample size of 120, it strongly suggests that there is sufficient variance and signal within this small sample to enable effective classification by ML models. If this observation holds, it implies that the small number of data points belonging to the "Disgust" class could still possess adequate discriminatory features. This supports the viability of continuing with the development and training of the ML models, perhaps with the aid of data augmentation or other techniques to mitigate the imbalance.

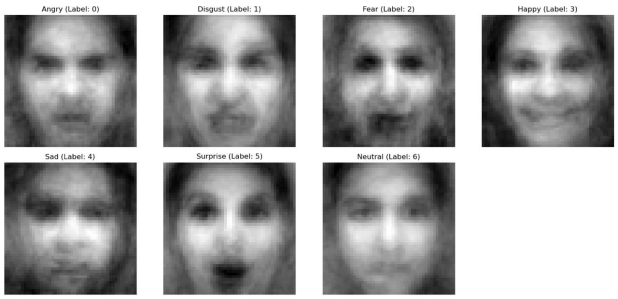

Figure 2: Averaged (element-wise) pixel values for the first 120 images grouped by class label

3 Data Pre-processing and Feature Extraction

3.1 Data Balancing

As discussed in section 2 a potential pre-processing step is addressing the data imbalance. Two simple techniques to address this imbalance are the oversampling of underrepresented class labels or the undersampling of overrepresented class labels. Oversampling runs the risk of overfitting, whereas undersampling reduces the amount of information available to train the model. Both techniques immediately limit the potential performance of the ML models. To achieve better performance, more sophisticated techniques can be used, which reduce overfitting while ensuring that all available information is being used. Three more sophisticated techniques are considered, namely: *class weighting*, *augmentation*, and *Synthetic Minority Oversampling Technique* or SMOTE [6].

3.1.1 Class Weighting

Class weighting is a technique in supervised learning classification where an imbalance among class labels can be addressed by adding a weighting during the optimization (assumed minimization) of the objective function, where under-represented class labels are given a greater weight, and over-represented class labels are given a smaller weight. This ultimately assumes the result of equality representation for the class labels; however, selecting a class weighting strategy is model-specific and is interdependent on other feature engineering decisions. By definition, non-objective-function-based models have a nontrivial process for implementing a class weighting strategy in the learning process. For models where class weighting is applicable, the chosen strategy for class weights can be treated as a hyper-parameter. It is only difficult to implement Class Weighting in MLP, hence it omitted for this model. [3]

3.1.2 Augmentation

Augmentation is a technique that involves the creation of artificial training data by applying sound transformations (or augmentations) onto the original training data to create a larger training data set; hence applying augmentation on the under-represented class labels can achieve class label balancing. Good domain understanding is required to understand the limitations of what transformations should be permitted. In the case of emotion classification from images, acceptable transformations would include mirroring the image or applying a small sheer or rotation to the image. Although augmentation is a form of oversampling, it reduces the risk of overfitting as it introduces potential feature diversity in the over-sampled data, promoting the learning of the general pattern instead of overfitting. Augmentation is applicable to any model as new training data is created unlike class weighting. [7]

3.1.3 SMOTE

SMOTE is a technique, that like augmentation, adds artificial training data to handle the imbalance in representation among the class labels. SMOTE shares the same benefits as augmentation however the creation of the new data happens by adding new data points

in the feature space constrained by k-nearest neighbors of like data points. It can be thought as taking a snapshot between the 'morph' or as the interpolation between identified like features. While it is possible to create new images when the 'feature space' is the pixels of the image this is far too computationally expensive, and is on the order of magnitude of training a model. Instead, data are created after feature extraction or reduction, where the feature space is reduced sufficiently to meet a good trade-off between computation complexity and the extent to which feature reduction is carried. The advantage of SMOTE over augmentation is that domain-specific knowledge of the underlying data is not required.

3.2 Feature Engineering

A processes of feature extraction and feature filtering are used to transform the raw input feature space (the image) to sufficiently representative, yet reduced feature set. The resulting feature set then undergoes feature-feature correctional filtering to remove redundant features. At later in the machine learning pipeline for further optimization filter-based methods
The size of the raw feature space of a 48×48 image is 2304 which is relatively significant. The SVM, MLP, Random Forrest, and Fuzzy Classifier models can all benefit with a smaller feature set size in terms of computational complexity. Some models like Fuzzy Classifier are sensitive to high-dimensionally or too large feature space and require the creation of *fuzzy sets* (to be discussed in a later section). For feature reduction of image-based data, it is important to perform feature extraction which preserves the spatial significance of the raw features or pixels. ML models can implicitly learn this significance; however, it comes at the cost of computational, model, and resource complexity. Another important requirement of feature selection is to minimize noise or those features that are less relevant to the task of classification (where the feature-target correlation is low) is true for SVM, MLP, and Fuzzy Classifier. A final consideration for the chosen feature extraction methods is that they would have to be relatively computationally efficient as they would be a part of the real-time classification pipeline; and would introduce additional processing latency.
Given the above considerations and requirements, the following two feature selection techniques are used: Histogram of Oriented Gradients (HOG), and Facial Landmark detection.

3.2.1 Histogram of Oriented Gradients

HOG is a feature extraction method that reduces feature space dimensionality by extracting the edges present in the image. It is also able to extract texture information in higher resolution images, but less less prevalent for the given 48×48 images. It works by computing 9 gradient intensities/scores for 8×8 cells, yielding $6 \times 6 = 36$ cells. Normalization is applied by scanning a 2×2 block over the cells. This results in $(5 \times 5) \times 36 = 900$ individual values encoding the gradient, texture and edge information of the image. This feature extraction method preserves the spatial significance of the image while reducing the size of the feature space. HOG can be computed fairly optimally due to its extensive use of basic linear algebra for which modern computing architectures are designed to handle. [8]

3.2.2 Facial Landmark Detection

Facial landmark detection is the localization of key facial landmarks within the image. For example, the pixel-based location and sizes of eyes, nose, mouth, cheeks, ears, chin, etc. in the image. Modern landmark detection libraries are able to identify 3-dimensional facial landmarks quite accurately and optimally using pre-trained landmark detection models. This project utilizes the *face mesh* landmark detection algorithm which provides 468 landmarks in a 3-dimensional space. The 2-dimensional projection (x and y coordinates to the picture plane) of these landmark points yields a feature space of $468 \times 2 = 936$ features. Many of these features will be redundant as the landmarks may be projected to the same pixel or neighboring pixels in the 48×48 image; however this will be drastically mitigated by applying a one-time variance and feature-feature correlation filtering. *Face mesh* utilities a ML model which has been highly optimized to run on mobile devices with limited computing power providing a high resolution baseline that offers flexibility at little computational cost. *Face mesh* was chosen over simpler landmark detection algorithms as it has the potential to be more beneficial with virtually insignificant complexity cost. [9]

4 Model Training

In this section, the training process of the different models: SVM, MLP, Random forest, and Fuzzy Classifier is explained. The selected models differ significantly in their approaches to 'learning' to achieve the ultimate goal of classification, and hence have varying requirements and hyper-parameter considerations. The nuances between the models are also discussed.
The various techniques of class balancing and the feature extraction techniques described in previous sections are applied to each of the models, and often the techniques that showed initial favorable utility were selected for the models.
It is important to note that the metric that was used when evaluating the usefulness of the above-mentioned techniques is the F1 model evaluation metric, as it is sensitive to poorly performing minority-classes in contrast to overall accuracy of a model alone. This metric was chosen because of the significance of the class imbalance in the data set.
In general a similar pipeline was applied to the SVM, MLP, and Random Forest models while conceptually different approach was applied to the fuzzy model.

4.1 SVM

The objective of a SVM in classification is to find a hyperplane (a high-dimension plane or boundary) that distinctly separates the data points of the different classes. The hyperplane is constructed by maximizing the distance between the training data points and the hyperplane. It can handle a degree of high dimensionality in the feature space. SVM can also identify non-linear patterns by using kernels; however, the kernel function is then considered a hyperparameter. Other hyper-parameters considered are C the marginal penalty, γ the kernel width. [3]

4.2 MLP

An MLP is forward-feeding series of nodes arranged in layers where the nodes between each layers are all interconnected with the previous layer through a non-linear *activation function*. The input nodes are where the raw data or feature space data is fed in to the model for classification purposes. The subsequent node layers are called the hidden layers, and finally there are output nodes. The number of output nodes are dictated by the number of classes. The major benefits of MLPs are that they find non-linear patterns and do not necessarily require feature selection.

The hyper parameters of an MLP are the shape of the hidden layers which describe how many nodes there are per layer as well as the number of layers. The general trend is that the larger the shape of the hidden layers the better the performance of the MLP; however, this comes at the cost of computational complexity in inference but especially in training. Naturally, the choice of activation function is another important hyperparameter, however *relu* or rectified linear unit is used for its effectiveness and computational efficiency. Other hyper-parmeters chosen are α the regularization strength which tunes the balance between overfitting and generalization; the learning rate and the batch size which introduce a trade-off between learning speed with the depth of optimization achievable (a true global minima versus a strong local minima). [3]

4.3 Random forest

A random forest is a collection or "forest" of decision trees. They are resistant to overfitting and can handle high dimensionality due to the fact that they implicitly perform feature selection by discarding redundant and noninformative features. The Random forest is versatile ML model.

The hyper parameters for random forests include the depth of individual trees as well as the number of trees in the forest. A larger and deeper forest is able achieve higher performance however at the cost of computational complexity in both training and inference. [3]

Table 2: Summary of selected pipeline methods for 'crisp' classifiers

Pipeline Method	Classifier Model		
	SVM	MLP	Random Forest
Feature Extraction			
HOG	Yes	Yes	Yes
Landmark Detection	Yes	Yes	Yes
Balancing Technique			
Augmentation	No	No	Used
SMOTE	Stronger	Stronger	Stronger
Class Weights	Weaker	Weaker	Weaker
Feature Filtering			
Feature-Feature Correlation	Yes	Yes	Yes
ANOVA F1	Marginally beter but more computationally expensive	Better, Selected	Not used
PCA Reduction	Selected	Sensitive to PCA variance compression	Not used

4.4 Model Training and Optimization Pipeline - Crisp' Classifiers

A general training and optimization approach was followed for the SVM, MLP, and Random forest. First, baseline hyper-parameters are used to identify which data balancing technique best improves model performance. It was found that *augmentation* did not benefit the SVM and MLP models, but did for the Random forest; as a result, a special training set was developed for Random forest that only augmented up to 100% of the minority classes (double the original number). Where applicable, SMOTE was selected to complete class balancing as it performed better than class weighting; even for MLP where no class weighting was performed.

Analysis of variance with the F1 metric filtering (ANOVA F1) and principal component analysis (PCA) selection was used to further minimize the feature set size before *grid search* hyper-parameter fine tuning, so as to minimize the time to run each hyper-parameter configuration in the grid search. The SVM model performed better with the features selected via ANOVA F1 compared to PCA however only marginally, hence PCA was selected due to its computational efficiency for the SVM model pipeline. Similarly ANOVA F1 outperformed PCA selection for the MLP model, this is possibly due to PCA over-compressing the variance which causes a less favorable result in MLP, hence ANOVA F1 is selected for the MLP pipeline. Due to the efficiency and self-feature filtering of training the Random forest model, no additional feature filtering was performed in its pipeline. Table 2 summaries the pipeline for the 'crisp' models

Hyper-parameter finetuning was performed on each model using the default scikit-learn values as a baseline and trying increments above and below to construct the grid search parameter grid. Refer to the notebook for the chosen hyper parameters.

4.5 Fuzzy classifier

A fuzzy classifier is a model that combines domain expertise and machine learning. It requires manual cre-

ation of *fuzzy sets* which follow logic grounded in solid domain knowledge. Instead the learning algorithm, finds patterns in interpreting and inferring truth based on the fuzzy features.
The specific model implemented was a Takagi-Sugeno Fuzzy Inference System (TSFIS), built using the pyFUME (Python Fuzzy Modelling environment) package. This model was trained on the combined HOG and Facial Landmark feature set, identical to the one used for the crisp classifiers. Given the high dimensionality of this feature space (1836 features), a key preprocessing step was the application of Principal Component Analysis (PCA) to reduce the data to its most significant components. The use of PCA priorotized the extraction of texture and shape patterns whilst keeping the computational complexity minimal. To address the class imbalance for this model, augmentation was used instead of SMOTE as the linear interpolation in SMOTE causes distortion after the PCA step.
To create a fuzzy set a *one-versus-the-rest* approach was followed to construct the fuzzy sets. This involves creating 7 fuzzy systems which act as binary classier - either identifying the target class or negatively identifying the 'rest'. This approach simplifies the construction of fuzzy sets for the underlying data, but does increase the model's computational and geneal complexity. However, the class size is only 7, meaning this increase in complexity is linear.
A tuning process was conducted to find the optimal hyper-parameters for the number of PCA components, the number of clusters (which form the basis of the fuzzy sets), and whether to apply SMOTE for class balancing.
[10]

5 Model Evaluation

5.1 SVM

The SVM model, tuned with C=2 and gamma=0.01 and applied to the SMOTE-balanced dataset, achieved an overall accuracy of 55.2%. Its performance is detailed in the confusion matrix in Figure 3.
The model's behavior is characterized by an extreme bias toward the "Happy" class, which achieved a 90.4% recall. While this demonstrates an effective identification of the majority class, it comes at a significant cost. Other emotions were frequently misclassified as "Happy," including 36.9% of "Sad" images and 26.6% of "Fear" images. Performance on minority classes was mixed; "Disgust" achieved a recall of 43.5%, but "Angry" (27.0% recall) and "Fear" (33.8% recall) were poorly identified.

5.2 MLP

The MLP model, which used balanced class weights as its balancing strategy, achieved an overall accuracy of 52.9%. As shown in Figure 4, its performance was more balanced than the SVM, with fewer egregious misclassifications into a single category.

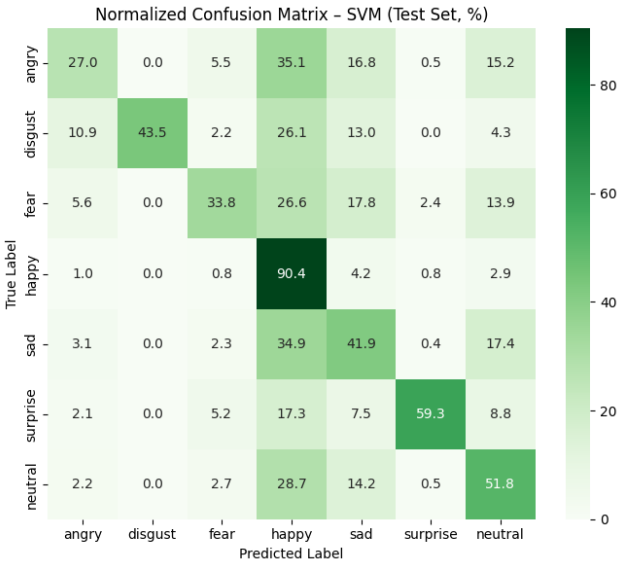

Figure 3: Normalized Confusion Matrix: SVM

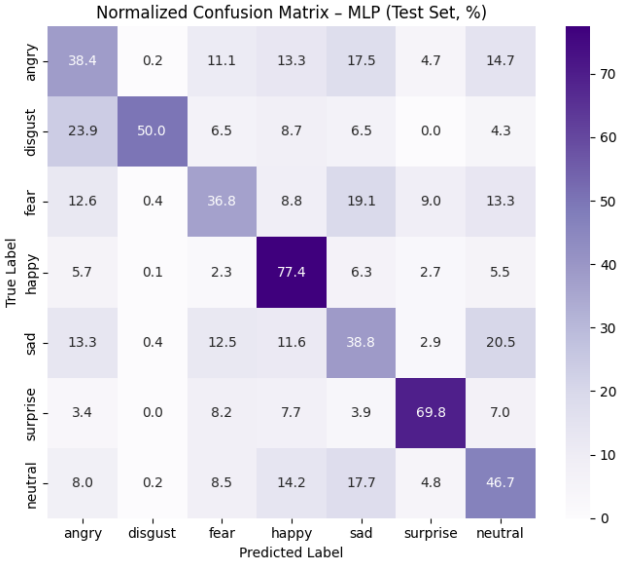

Figure 4: Normalized Confusion Matrix: MLP

The "Happy" class was still the best-performing (76.4% recall), but not to the detriment of all other classes. This model's strategy yielded a slightly better recall for "Disgust" (47.8%) and "Angry" (38.4%) compared to the SVM. However, it demonstrated notable confusion between "Sad" and "Neutral," misclassifying 24.3% of "Sad" images as "Neutral."

5.3 Random Forest

The Random Forest model presents the most promising results, particularly in the context of the class imbalance problem. While its full classification report was not captured in the execution logs, its confusion matrix (Figure 5) illustrates a strong, balanced performance. This model is the only one to successfully address the challenge of the "Disgust" class, achieving a recall of approximately 50%. This success is a direct result of

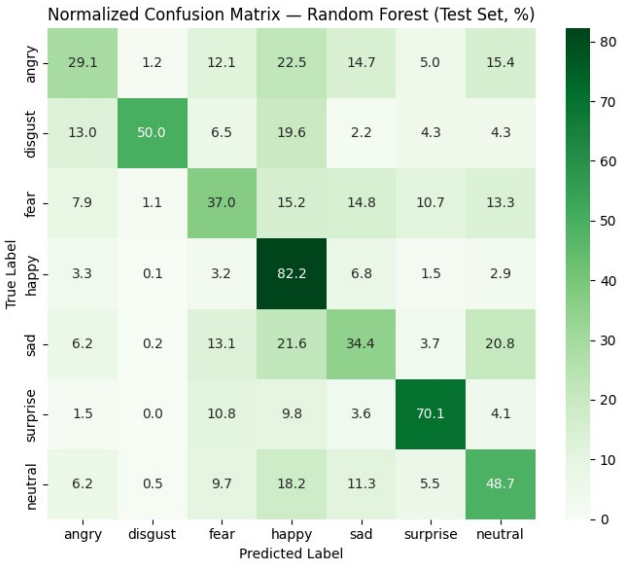

Figure 5: Normalized Confusion Matrix: Random Forest

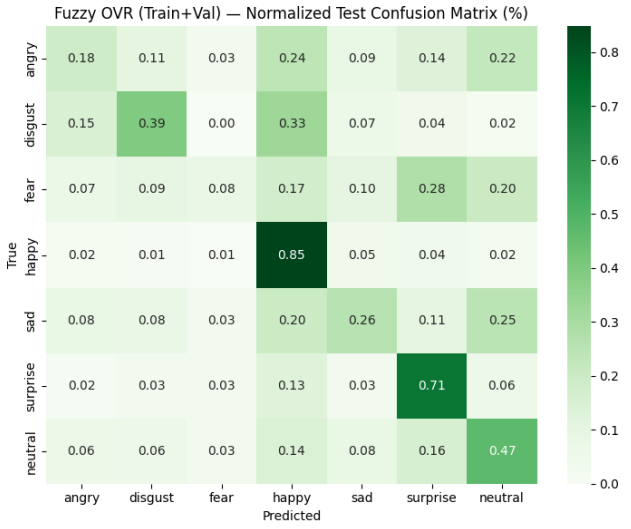

Figure 6: Normalized Confusion Matrix: Fuzzy Classifier

its unique pipeline, which was the only one to combine both data augmentation (via image rotation and flipping) and SMOTE balancing. This dual strategy created a more robust and diverse set of training examples for the minority classes. The model also achieved the highest recall for "Surprise" (70.1%) and performed well on "Happy" (82.2%), indicating it is the most well-rounded of the crisp classifiers.

5.4 Fuzzy Classifier

The final Fuzzy Classifier's confusion matrix is shown in figure 6. It can be seen that the model performs relatively well for correctly classifying "happy" and "surprise", however it performs poorly for the rest. This imbalance is present in the F1-score of 0.47. This result could be caused by the class imbalance despite the augmentation in addition to the pattern quality of the images in other classes. While the one-versus-the-rest approach as proven to be a viable method to implement a Fuzzy Classifier, a more domain-knowledge-driven approach might yield better results and handle the inconsistencies in the data.

6 Reflection on Demo Implementation

A real-time demo was created to validate the model in a live setting. The application, built with OpenCV, captures video frames from a webcam and processes them in a sequential pipeline. First, a Haar cascade classifier detects faces within the frame. This detected region is then cropped, resized to the required 48×48 pixel dimensions, and converted to grayscale. On this processed image, the HOG feature extraction is applied, and the resulting feature vector is fed into the trained SVM model for the final emotion prediction. This section provides a qualitative evaluation of the demo, the challenges in its implementation, and an analysis of the gap between the test dataset and a real-world camera feed. The implementation of this real-time pipeline presented two significant technical challenges: latency and detection stability. The full, five-step process—from capture to prediction—is computationally expensive. Running this entire pipeline on every single frame introduced a noticeable lag, resulting in a delay between the user's expression and the displayed prediction, which made the demo feel unresponsive. Compounding this was the issue of detection stability. The Haar cascade classifier proved sensitive to environmental factors. Suboptimal lighting, non-frontal head poses, or partial occlusions (e.g., from glasses or hands) would cause the detector to fail, breaking the entire pipeline. This manifested as a flickering prediction that would disappear or freeze on the last valid detection. In terms of qualitative performance, the demo's behavior in a real-world scenario is a direct reflection of the SVM's quantitative results from the test set. The model is highly effective and reliable at identifying "Happy" expressions, which aligns perfectly with the 90.4% recall observed for that class. It also performs reasonably well when presented with clear "Surprise" and "Neutral" faces. However, the demo also inherits the SVM's significant weaknesses. It visibly struggles to consistently identify "Angry," "Fear," or "Sad" expressions. The SVM's strong bias toward the majority class, as seen in the confusion matrix, is also evident in the demo, with many subtle expressions or even neutral faces being frequently misclassified as "Happy." This confirms that the model's high recall for "Happy" was achieved at the cost of high false positives from other classes. The model's performance on the live camera feed is noticeably poorer than on the provided dataset, and this discrepancy highlights a significant domain gap between the training data and a real-world application. The FER2013 dataset consists of static, low-

resolution (48×48), grayscale, and mostly pre-aligned facial images. In contrast, the demo must process a high-resolution color video feed in real-time, introducing motion blur and scaling artifacts not present in the training data. Furthermore, the dataset is "clean" of environmental variables, whereas the live feed suffers from highly variable lighting and shadows, which drastically affect HOG features. Perhaps the most critical difference, however, is the nature of the expressions themselves. The dataset contains posed, static, and often exaggerated expressions. The model has learned to classify these specific poses. In the real world, expressions are dynamic, subtle, and involve motion. The model was not trained on this dynamic data, so it fails to capture the nuances of a natural expression and instead seems to wait for the user to replicate a specific static pose from the training set.

References

[1] OpenCV, "Open Source Computer Vision Library." http://opencv.org/, 2015.

[2] D. E. King, "Dlib-ml: A Machine Learning Toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] Kaggle, "Facial Expression Recognition (FER2013) Dataset." https://www.kaggle.com/datasets/deadskull7/fer2013, 2013. Accessed: October 6, 2025.

[5] I. J. Goodfellow, D. Erhan, P.-L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, Y. Zhou, C. Ramaiah, F. Feng, R. Li, X. Wang, D. Athanasakis, J. Shawe-Taylor, M. Milakov, J. Park, R. Ionescu, M. Popescu, C. Grozea, J. Bergstra, J. Xie, L. Romaszko, B. Xu, Z. Chuang, and Y. Bengio, "Challenges in Representation Learning: A report on three machine learning contests," *Neural Networks*, vol. 64, pp. 59–63, 2015. Special Issue on "Deep Learning of Representations".

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, June 2002.

[7] A. Jung, "imgaug: Image augmentation for machine learning experiments." https://github.com/aleju/imgaug, 2020.

[8] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: Image processing in Python," *PeerJ*, vol. 2, p. e453, 2014.

[9] Google, "Mediapipe face mesh." https://developers.google.com/mediapipe/solutions/vision/face_landmarker, 2023. Accessed: 2025-10-24.

[10] C. Fuchs, S. Spolaor, M. S. Nobile, and U. Kaymak, "pyFUME: a Python package for fuzzy model estimation," in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–8, IEEE, 2020.