Comprehensive Facial Expression Recognition on FER2013: Face Alignment, Sampling Strategies, and Real-Time Deployment

Utku Murat ATASOY
Artificial Intelligence Engineering
TOBB University of Economics and Technology
u.atasoy@etu.edu.tr

***Abstract*—In this project, the multi-class facial expression recognition problem was addressed using the FER2013 dataset. Significant class imbalances within the dataset, particularly the low representation of the *Disgust* class, negatively impact model performance. To mitigate this, three different sampling strategies (*undersampling*, *oversampling*, *hybrid sampling*) were employed along with various loss functions (Cross Entropy, Class Weighted, Focal Loss). Transfer learning approaches were utilized to train and compare XceptionNet, MobileNet-V3-Large-100, EfficientNet-B0, and ResNet-18 architectures. For the best-performing combination, XceptionNet + Cross Entropy Loss + Hybrid Sampling (Target=8989), face alignment was performed using the FaceNet MTCNN structure, which significantly improved class-based performances. The resulting models were converted to ONNX format, and real-time performance tests showed that the system could operate with 66.76% accuracy and low latency.**

***Index Terms*—Facial Expression Recognition, FER2013, data imbalance, transfer learning, sampling strategies, deep learning.**

I. Introduction

Facial Expression Recognition (FER) has important applications in various fields such as human-machine interaction, security, driver behavior analysis, education, and healthcare. Rapid advances in image processing and deep learning techniques have made it possible to automatically and accurately classify facial expressions. However, class imbalances encountered in datasets (for example, the small number of samples in the *Disgust* class) adversely affect model performance.

Within the scope of this study, an approach to the multi-class facial expression recognition problem was developed; the goal was to build a balanced and high-performing model encompassing all emotion classes. Both numerical and visual quality balancing processes were applied to the data; in particular, the face alignment step was performed automatically and consistently using the MTCNN algorithm.

Different sampling and loss strategies were tested on transfer learning-based pre-trained deep learning models (CNN architectures); the best result was achieved using the XceptionNet model with a hybrid sampling and Cross Entropy Loss combination. The model's class-based performance metrics (Precision, Recall, F1 Score, and Accuracy) were evaluated, and it was also tested in real-time applications after being converted to ONNX format. During testing, XceptionNet achieved **Accuracy = 66.76%** on the test set after alignment, and an average of approximately 140 FPS was observed in real-time tests with ONNX models.

The developed codes and detailed presentation materials related to the project can be found in the reference section of the report ([1], [2]).

II. Literature Review

When previous studies on facial expression recognition (FER) are examined, it is seen that the FER2013 dataset is commonly used, and that transfer learning methods are frequently preferred. One frequently encountered approach in the literature is the exclusion of underrepresented classes such as *Disgust* from analysis. Although excluding this class can increase model accuracy, it offers a limited solution for real-world applications.

Khaireddin and Chen [3] demonstrated in their comparative analysis using various transfer learning models on FER2013 that including the *Disgust* class reduced accuracy rates. Similarly, Kusuma and Lim [4] evaluated the effect of fine-tuning with the VGG-16 model and emphasized that data imbalance posed a significant issue. Ogui and colleagues [5] revealed the advantages of hybrid architectures in real-time facial expression recognition systems, while Zahara and her team [6] analyzed the performance of CNN-based models in micro-expression detection on low-powered devices.

Recent literature shows that MTCNN and FaceNet-based approaches have become prominent in face detection and alignment steps. Qi and colleagues [7] utilized the P-Net, R-Net, and O-Net structures of MTCNN to extract face regions and performed vector-based face comparisons through FaceNet, achieving 86% accuracy in their tasks. Wu and Zhang [8] noted that traditional algorithms (AdaBoost, Haar-like, DPM) were both slower and less accurate, and showed that MTCNN and FaceNet could reach up to 99.85% accuracy in real-time systems. Roy and colleagues [9] introduced a model that achieved 99.87% accuracy in attendance monitoring systems using MTCNN-supported FaceNet structures.

In our project, in accordance with these literature methods, face alignment was performed using the MTCNN architecture, enabling healthier learning by the model. Furthermore, post-training models were converted into ONNX format and

tested in real-time, fully replicating live application scenarios discussed in the literature. In this context, a comprehensive system was presented by combining multiple approaches from the literature within a single project.

The table below summarizes the accuracy results of some literature studies depending on whether the *Disgust* class was included:

Method	Data with Disgust	Data without Disgust
Transfer Learning using Xception	0.4078	0.4149
Transfer Learning + Fine Tuning using Xception	0.6485	0.6517
Transfer Learning using ResNet152V2	0.3798	0.3898
Transfer Learning + Fine Tuning using ResNet152V2	0.5551	0.2714
Transfer Learning using MobileNetV2	0.4013	0.3941
Transfer Learning + Fine Tuning using MobileNetV2	0.4771	0.4675
Transfer Learning using EfficientNetB0	0.4084	0.4122
Transfer Learning + Fine Tuning using EfficientNetB0	0.5856	0.5961
Transfer Learning using InceptionResNetV2	0.4153	0.4235
Transfer Learning + Fine Tuning using InceptionResNetV2	0.4138	0.4154
Self Made CNN	0.5857	0.5888

This table clearly shows that overall accuracy tends to decrease when the Disgust class is included. However, in real-world scenarios, it is not feasible to ignore such expressions. Therefore, in our study, all classes were included in the model, and to overcome data imbalance, sampling strategies, weighted/focal loss applications, and face alignment methods were employed to increase performance.

Moreover, open-source projects [10], [11] were examined in terms of both architectural choices and application examples, and some implementation practices were integrated into this project.

III. Dataset, Data Properties, and Features

This section presents detailed information on the source of the FER2013 dataset used in the project, feature descriptions, data preprocessing steps, data imbalance, and feature explanations. Additionally, normalization of raw data, face alignment operations (based on FaceNet MTCNN), and analyses conducted for model explainability (XAI – LIME) are described here.

A. Data Source and Dataset Definition

The dataset used in the project is **FER2013**, an important source containing grayscale facial images of 48x48 pixels, accessible via Kaggle. The dataset includes seven emotion classes (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral). For detailed information and download link, click here.

Fig. 1. Example image from the FER2013 dataset.

B. Dataset Size and Class Distribution

This dataset contains a total of **35,887** facial images and is particularly notable for the imbalance among classes. For detailed information and download link, click here.

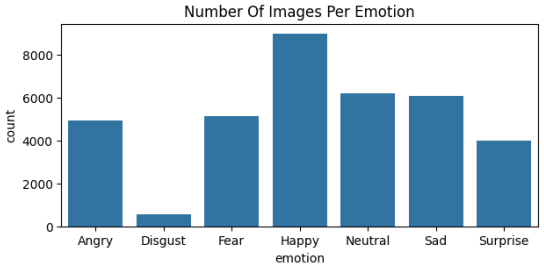

Fig. 2. Data distribution in the raw (unprocessed) dataset.

The data contains samples from 7 different emotions, distributed as follows:

- **Happy:** 8989 samples
- **Neutral:** 6198 samples
- **Sad:** 6077 samples
- **Fear:** 5121 samples
- **Angry:** 4953 samples
- **Surprise:** 4002 samples
- **Disgust:** 547 samples

This distribution, especially the very low number of samples in the *Disgust* class compared to other classes, causes imbalance issues during model training. The low number of examples for the *Disgust* class has led the model to insufficiently focus on this emotion during training, resulting in lower performance. In the literature, studies that achieve high accuracy often exclude the *Disgust* class and proceed with six classes.

In this project, despite class imbalance, it was aimed to achieve high accuracy by including the *Disgust* class. To this end, various sampling strategies were applied:

- **Undersampling:** Random samples from over-represented classes were selected to reduce class imbalance in the training data. This prevented the model from being biased toward frequent classes.
- **Oversampling:** The underrepresented *Disgust* class was augmented, allowing the model to learn this class better and thus improve class balance.

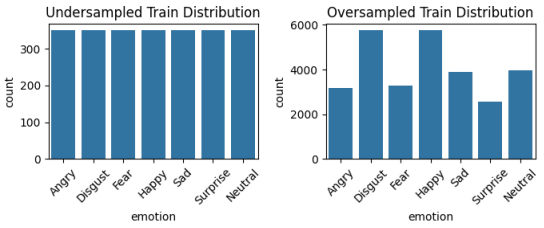

Fig. 3. Class distributions after undersampling and oversampling applications.

- **Hybrid Sampling:** By combining oversampling and undersampling methods, the number of examples for all classes was balanced to a specific target value. In the first strategy, all classes were balanced to approximately 3000 samples; in the second strategy, they were balanced to 8989 samples.

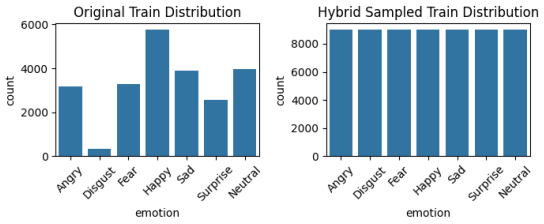

Fig. 4. After hybrid sampling, all classes were balanced.

C. Data Preprocessing, Normalization, and Feature Explanations

All data were processed using the following preprocessing steps:

- **Normalization:** Raw images were normalized from pixel values ranging 0–255 to a range of 0–1.
- **Image Processing:** Before face alignment using MTCNN, each image was converted to RGB format and resized to 96x96 dimensions.
- **Data Augmentation:** Data diversity was increased by applying rotation, cropping, horizontal flipping, brightness, and contrast adjustments.

Features represent the numerical pixel values in each image, while labels are nominal (unordered, multiclass) categorical data. In particular, the underrepresentation of the *Disgust* class makes it harder for the model to learn effectively.

D. Model Interpretability with Explainable AI (XAI): LIME

To better understand the decision-making mechanisms of models trained without face alignment and only using sampling methods, the **LIME (Local Interpretable Model-agnostic Explanations)** method was applied. With this method, the regions of images contributing positively or negatively to model predictions were visualized with color codes.

Specifically:

- **Angry:** The model misclassified unclear facial expressions as *Sad* due to shadowed areas and sunglasses obscuring the eyes. Positive contributions were concentrated around the nose and sunglasses.
- **Disgust:** Positive contributions appeared around the nose, forehead wrinkles, and between the eyebrows. However, as these areas overlapped with *Angry* expressions, the model sometimes misclassified.
- **Surprise:** Although mouth opening and eye areas are important for *Surprise*, the model confused these with *Sad* and produced low-weight positive contributions. Contributions were more spread out.
- **Neutral:** Due to the symmetric and relaxed muscle structure, the model associated soft curves around cheeks and eyes with the *Happy* class, causing misclassification.

The following figures show examples from LIME analysis:

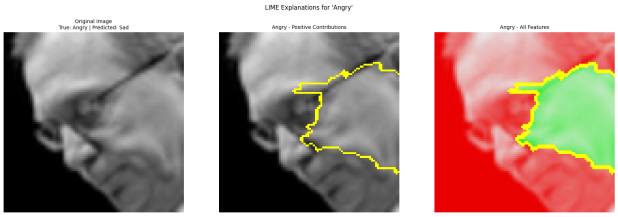

Fig. 5. LIME explanation for an **Angry** class example. Although the true label was *Angry*, the model predicted *Sad*. Positive contributions are concentrated around the eyes and nose.

Fig. 6. LIME explanation for a **Disgust** class example. Although the true label was *Disgust*, the model predicted *Angry*. Positive contributions are around the nose, forehead, and eyebrows.

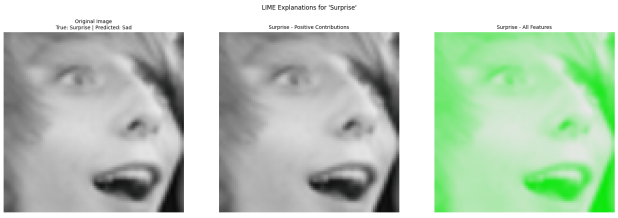

Fig. 7. LIME explanation for a **Surprise** class example. Despite clear mouth and eye regions, the model predicted *Sad*. Green areas indicate positive contributions.

E. Face Alignment Operations and FaceNet MTCNN Application

Among the models obtained by experimenting with sampling and different loss functions, the most successful combination — **XceptionNet + Cross Entropy Loss (Hybrid Sampling, Target = 8989)** — was analyzed in terms of interpretability using the **LIME (Local Interpretable Model-agnostic Explanations)** method. LIME allows visualizing which parts of an image the model focuses on while making predictions, making it possible to analyze the reasons behind misclassifications.

As observed in the LIME visualizations, in some images, the face region is not fully visible, and sometimes the model focuses on areas outside the face. This leads the model to make incorrect predictions and negatively impacts training. To

address this issue, face alignment (*face alignment*) was applied to the dataset.

In this operation, the **FaceNet MTCNN (Multi-task Cascaded Convolutional Networks)** was used. MTCNN uses a three-stage structure (P-Net, R-Net, O-Net) to detect and align faces within an image:

- **P-Net:** Detects candidate face regions.
- **R-Net:** Filters candidate regions to produce more precise face locations.
- **O-Net:** Determines the final face bounding boxes and the five key points of the face (eyes, nose tip, mouth corners).

In the project, this structure was used to obtain **96x96 pixel** aligned facial images. The operation was performed in batch mode on a GPU for significant speed advantages.

Each sample underwent the following alignment steps:

1) The 48x48 grayscale image was converted to RGB format.
2) The face region was detected using MTCNN and normalized to 96x96 size.
3) Samples without detectable faces (optionally based on a parameter) were removed from the dataset.

Below are some summary statistics regarding the alignment process:

- **Training set:** 62,923 samples → 10,463 faces could not be detected → **52,460** samples remained.
- **Validation set:** 5,742 samples → 928 faces could not be detected → **4,814** samples remained.
- **Test set:** 7,178 samples → 1,188 faces could not be detected → **5,990** samples remained.

Fig. 8. (a) Samples **without** face alignment.

Fig. 9. (b) Samples **aligned** with FaceNet MTCNN.

Fig. 10. The effect of face alignment on the data. (a) Without alignment, variations in head angle, position, and framing are visible; (b) With alignment, faces are centered, properly scaled, and more symmetrical. This provides a major advantage for model training and performance.

These results demonstrate that in a significant portion of the FER2013 dataset, faces were either not directly detectable by the model or were misaligned. Therefore, this preprocessing step significantly improved the model's overall accuracy and stability.

F. The Impact of Face Alignment on LIME Explanations

To more concretely demonstrate the impact of face alignment on the model's decision-making process, old (Figures 5, 6, and 7) and new LIME explanations were compared.

In models without alignment, it was observed that in the LIME explanations, the areas the model focused on were limited and sometimes included regions outside the face. For instance, in Figure 5, for an example of the **Angry** class, the model focused only on small areas like around the eyes and nose; in Figure 6, although there was some correct focus for a **Disgust** expression, the model predicted *Sad*. In Figure 7, despite the mouth and eyes being distinct, the classification was again incorrect.

After applying face alignment (Face Alignment), it was observed that the model's attention was more evenly and meaningfully spread across the entire facial region. As shown in Figures 11, 12, and 13, the model now considers all critical areas that determine expressions — eyebrows, eyes, nose, and mouth — in a more balanced manner and properly evaluates non-facial areas as having negative contributions.

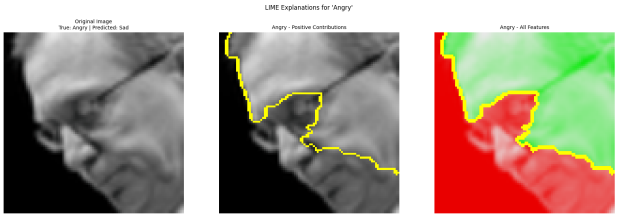

Fig. 11. LIME explanation for an **Angry** class example after face alignment. The model now focuses on a broader area of the face.

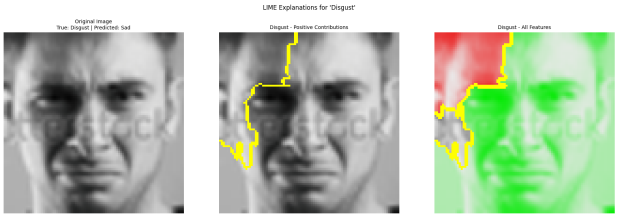

Fig. 12. LIME explanation for a **Disgust** class example after face alignment. Balanced attention around the nose, forehead, and eyebrows directs the model's focus more correctly.

Fig. 13. LIME explanation for a **Surprise** class example after face alignment. More balanced attention to the key areas of the expression can be observed.

These results show that face alignment not only increases the model's accuracy but also makes its decision-making mechanism more reliable.

IV. Models Used

In this project, four different pre-trained CNN architectures based on the transfer learning approach were used for the

multi-class facial expression recognition problem. The training, validation, and test splits were determined as 70%, 10%, and 20%, respectively. Each model was fine-tuned to improve the results on the FER2013 dataset compared to similar studies in the literature.

A. Model Selection and Descriptions

- **XceptionNet:** A deep, modular architecture capable of superior feature extraction on large datasets. In the literature, some of the highest accuracies in FER problems have been reported with fine-tuning on XceptionNet. In this project, it provided the best results with a hybrid sampling strategy (Target = 8989) and Cross Entropy Loss.
- **MobileNet-V3-Large-100:** A lightweight model optimized for mobile devices with low computational power requirements, making it suitable for real-time applications. However, some complex expression separations showed lower performance compared to XceptionNet.
- **EfficientNet-B0:** Optimized for efficiency and parameter count, EfficientNet-B0 offers advantages in scalability and fast inference times. In our project, its general model performance and FPS/latency results in real-time applications were evaluated.
- **ResNet-18:** A simpler and lighter architecture that minimizes gradient loss through residual connections in deep networks. While offering consistent and stable results, it slightly lagged behind other models in terms of overall accuracy.

B. Training and Evaluation Process

All models were trained on the FER2013 dataset with a 70% training, 10% validation, and 20% test split. Within the scope of the multi-class classification problem, each model's performance was evaluated using Precision, Accuracy, Recall, and F1 Score metrics.

During training, to address data imbalance, undersampling, oversampling, and hybrid sampling strategies were applied. Additionally, different loss functions such as Cross Entropy, Class Weighted, and Focal Loss were experimented with. Thus, it was aimed that models could perform better even on challenging classes like the underrepresented *Disgust* class.

C. Justification for Model Selection

In the literature, it has been reported that transfer learning-based models achieve high success in FER problems, and pre-trained networks provide superior generalization ability, especially when data quantity is limited [3], [4], [5].

- **XceptionNet** was preferred because of its ability to extract detailed features in complex expressions.
- **MobileNet-V3-Large-100** and **ResNet-18** were chosen for real-time applications and scenarios requiring low computational power.
- **EfficientNet-B0** offered an advantage in practical use due to its optimized parameter count, providing both high accuracy and fast inference times.

These model selections allowed the development of strategies aimed at increasing the applicability of the project in real-world scenarios, based on both the training results obtained and comparisons with reported literature values.

V. Test Results

In the project, the training process was meticulously examined by using different sampling strategies and loss functions. Below are the project plan, flow chart, experimental results, and model test outputs.

The main stages of the project are summarized in Table I.

TABLE I
PROJECT STAGES AND DESCRIPTIONS

Stage	Description
Dataset Acquisition	The FER2013 dataset was obtained from Kaggle. It contains 48x48 pixel grayscale facial images and seven emotion classes.
Data Preprocessing	Images were normalized from raw data and resized to 48x48; data augmentation methods such as rotation, cropping, brightness, and contrast adjustments were applied.
Sampling Strategies	To mitigate class imbalance, undersampling, oversampling, and hybrid sampling methods were applied.
Loss Functions	During model training, different loss functions such as Cross Entropy, Class Weighted, and Focal Loss were used to minimize the effects of data imbalance.
Model Training	Using the transfer learning approach, fine-tuning was performed on XceptionNet, MobileNet-V3-Large-100, EfficientNet-B0, and ResNet-18 models. Data splits were made as 70% training, 10% validation, and 20% testing.
Performance Evaluation	After training, models were evaluated using Precision, Accuracy, Recall, and F1 Score metrics; confusion matrices and class-based classification reports were also generated.
Model Interpretability via XAI	Using the LIME method, the areas where the model focused during prediction were visualized, and the reasons behind misclassifications were analyzed.
Face Alignment and Retraining	Based on findings from LIME analyses, face alignment was performed using the FaceNet MTCNN architecture. Models were retrained on the aligned dataset, resulting in significant improvements in class-based performance.
Testing	Retrained models were evaluated on the test set. Overall performance and class-based performances were analyzed in detail.
Real-Time Application	The best models were converted into ONNX format and tested on live video streams, performing face detection via FaceNet MTCNN, preprocessing, and softmax classification in real-time. The system was optimized to operate with low latency and high FPS.

A. Obtained Results

1) Without Applying Sampling (Baseline): Table II presents the performance metrics of the models trained without applying any sampling strategy.

TABLE II
PERFORMANCE METRICS OBTAINED WITHOUT APPLYING SAMPLING

Model Name	Loss Type	Epoch	Batch Size	Precision	Accuracy	Recall	F1 Score
XceptionNet	Cross Entropy	30	64	0.6692	0.6708	0.6708	0.6692
MobileNet-V3-Large-100	Cross Entropy	30	64	0.6441	0.6461	0.6461	0.6442
EfficientNet-B0	Cross Entropy	30	64	0.6442	0.6442	0.6442	0.6436
ResNet-18	Cross Entropy	30	64	0.6437	0.6454	0.6454	0.6431
XceptionNet	Class Weighted	30	64	0.6590	0.6602	0.6602	0.6588
MobileNet-V3-Large-100	Class Weighted	30	64	0.6252	0.6301	0.6301	0.6263
EfficientNet-B0	Class Weighted	30	64	0.6365	0.6369	0.6369	0.6365
ResNet-18	Class Weighted	30	64	0.6330	0.6339	0.6339	0.6322
XceptionNet	Focal Loss	30	64	0.6568	0.6565	0.6565	0.6556
MobileNet-V3-Large-100	Focal Loss	30	64	0.6315	0.6282	0.6282	0.6292
EfficientNet-B0	Focal Loss	30	64	0.6259	0.6269	0.6269	0.6263
ResNet-18	Focal Loss	30	64	0.6227	0.6265	0.6265	0.6224

2) When Undersampling is Applied: Table III shows the results obtained using the undersampling strategy. In this method, it was observed that excessively reducing the amount of data negatively affected model performance.

TABLE III
PERFORMANCE METRICS OBTAINED WITH UNDERSAMPLING

Model Name	Loss Type	Epoch	Batch Size	Precision	Accuracy	Recall	F1 Score
XceptionNet	Cross Entropy	30	64	0.5334	0.5199	0.5199	0.5218
MobileNet-V3-Large-100	Cross Entropy	30	64	0.4231	0.3831	0.3831	0.3967
EfficientNet-B0	Cross Entropy	30	64	0.4287	0.4061	0.4061	0.4131
ResNet-18	Cross Entropy	30	64	0.5139	0.4858	0.4858	0.4951
XceptionNet	Class Weighted	30	64	0.5077	0.4784	0.4784	0.4872
MobileNet-V3-Large-100	Class Weighted	30	64	0.3885	0.3150	0.3150	0.3336
EfficientNet-B0	Class Weighted	30	64	0.3892	0.3313	0.3313	0.3476
ResNet-18	Class Weighted	30	64	0.4195	0.2751	0.2751	0.2958
XceptionNet	Focal Loss	30	64	0.5136	0.4827	0.4827	0.4914
MobileNet-V3-Large-100	Focal Loss	30	64	0.3679	0.2888	0.2888	0.3088
EfficientNet-B0	Focal Loss	30	64	0.3722	0.3218	0.3218	0.3352
ResNet-18	Focal Loss	30	64	0.4270	0.2871	0.2871	0.3155

3) When Disgust Class Oversampling is Applied: Table IV presents the results obtained by increasing the number of examples specifically for the underrepresented *Disgust* class through oversampling.

TABLE IV
PERFORMANCE METRICS OBTAINED WITH OVERSAMPLING (DISGUST CLASS)

Model Name	Loss Type	Epoch	Batch Size	Precision	Accuracy	Recall	F1 Score
XceptionNet	Cross Entropy	30	64	0.6599	0.6627	0.6627	0.6605
MobileNet-V3-Large-100	Cross Entropy	30	64	0.6347	0.6400	0.6400	0.6369
EfficientNet-B0	Cross Entropy	30	64	0.6296	0.6339	0.6339	0.6310
ResNet-18	Cross Entropy	30	64	0.6366	0.6407	0.6407	0.6377
XceptionNet	Class Weighted	30	64	0.6528	0.6580	0.6580	0.6539
MobileNet-V3-Large-100	Class Weighted	30	64	0.6266	0.6241	0.6241	0.6240
EfficientNet-B0	Class Weighted	30	64	0.6270	0.6291	0.6291	0.6276
ResNet-18	Class Weighted	30	64	0.6219	0.6232	0.6232	0.6216
XceptionNet	Focal Loss	30	64	0.6569	0.6555	0.6555	0.6559
MobileNet-V3-Large-100	Focal Loss	30	64	0.6181	0.6197	0.6197	0.6177
EfficientNet-B0	Focal Loss	30	64	0.6264	0.6254	0.6254	0.6242
ResNet-18	Focal Loss	30	64	0.6193	0.6154	0.6154	0.6156

4) When Hybrid Sampling (Target = 3000) is Applied: Table V shows the results obtained by balancing all classes to approximately 3000 examples using hybrid sampling.

TABLE V
PERFORMANCE METRICS OBTAINED WITH HYBRID SAMPLING (TARGET = 3000)

Model Name	Loss Type	Epoch	Batch Size	Precision	Accuracy	Recall	F1 Score
XceptionNet	Cross Entropy	30	64	0.6487	0.6439	0.6439	0.6450
MobileNet-V3-Large-100	Cross Entropy	30	64	0.6232	0.6134	0.6134	0.6174
EfficientNet-B0	Cross Entropy	30	64	0.6223	0.6165	0.6165	0.6187
ResNet-18	Cross Entropy	30	64	0.6245	0.6226	0.6226	0.6231
XceptionNet	Class Weighted	30	64	0.6509	0.6357	0.6357	0.6409
MobileNet-V3-Large-100	Class Weighted	30	64	0.6002	0.5952	0.5952	0.5971
EfficientNet-B0	Class Weighted	30	64	0.6088	0.5957	0.5957	0.6007
ResNet-18	Class Weighted	30	64	0.5974	0.5910	0.5910	0.5916
XceptionNet	Focal Loss	30	64	0.6556	0.6521	0.6521	0.6525
MobileNet-V3-Large-100	Focal Loss	30	64	0.6232	0.6134	0.6134	0.6174
EfficientNet-B0	Focal Loss	30	64	0.6219	0.6163	0.6163	0.6184
ResNet-18	Focal Loss	30	64	0.6278	0.6293	0.6293	0.6280

5) When Hybrid Sampling (Target = 8989) is Applied: Table VI shows the results where all classes were oversampled to match the size of the Happy class (8989 samples).

TABLE VI
PERFORMANCE METRICS OBTAINED WITH HYBRID SAMPLING (TARGET = 8989)

Model Name	Loss Type	Epoch	Batch Size	Precision	Accuracy	Recall	F1 Score
XceptionNet	Cross Entropy	30	64	0.6630	0.6668	0.6668	0.6636
MobileNet-V3-Large-100	Cross Entropy	30	64	0.6377	0.6402	0.6402	0.6378
EfficientNet-B0	Cross Entropy	30	64	0.6332	0.6351	0.6351	0.6326
ResNet-18	Cross Entropy	30	64	0.6298	0.6293	0.6293	0.6290
XceptionNet	Class Weighted	30	64	0.6523	0.6545	0.6545	0.6523
MobileNet-V3-Large-100	Class Weighted	30	64	0.6295	0.6291	0.6291	0.6289
EfficientNet-B0	Class Weighted	30	64	0.6247	0.6241	0.6241	0.6229
ResNet-18	Class Weighted	30	64	0.6314	0.6262	0.6262	0.6283
XceptionNet	Focal Loss	30	64	0.6535	0.6500	0.6500	0.6508
MobileNet-V3-Large-100	Focal Loss	30	64	0.6298	0.6271	0.6271	0.6275
EfficientNet-B0	Focal Loss	30	64	0.6190	0.6213	0.6213	0.6194
ResNet-18	Focal Loss	30	64	0.6316	0.6269	0.6269	0.6284

6) Comparison and Best Combination Analysis: As observed, the combination of **Hybrid Sampling (Target=8989)** with **Cross Entropy Loss** and the **XceptionNet** model produced the best results. It achieved an accuracy of **66.68%** on the test set before face alignment was applied.

This result was further improved by conducting face alignment (discussed in later sections), leading to even higher real-world usability in real-time applications.

VI. REAL-TIME TESTING AND ONNX PERFORMANCE COMPARISON

After completing the training process, models were evaluated for their usability in real-time facial expression recognition applications by converting them to the **ONNX (Open Neural Network Exchange)** format. ONNX is an open format that enables interoperability between different frameworks and works efficiently with low-level inference engines (e.g., ONNX Runtime, TensorRT).

Below, first the real-time performance metrics for normal ONNX models, and then for **simplified** ONNX models, are presented, followed by an analysis of the obtained results.

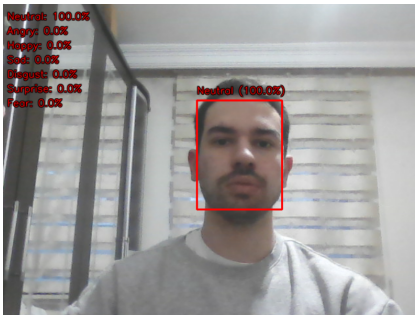

Fig. 14. A screenshot from the real-time facial expression recognition application.

As seen in Figure 14, the developed system processes live camera feeds, detects face regions using **MTCNN**, and predicts the corresponding emotion class in real-time by applying preprocessing and softmax classification.

The real-time application flow includes:

- Face detection on frames captured from the camera using **MTCNN**.
- Cropping, normalizing, and resizing the detected face to **96x96**.

- Feeding the processed face image as input to the ONNX model.
- Applying softmax to the model output to predict the most probable emotion class.
- Displaying the predicted emotion label alongside the detected face on the screen.

Thanks to these optimizations, the system operates very smoothly, performing inference dozens of times per second.

1) Real-Time Performance of Normal ONNX Models: After exporting the models from PyTorch to ONNX using a dummy input of size 96x96, their real-time performance was measured. Table VII shows the performance metrics for normal ONNX models:

TABLE VII
REAL-TIME PERFORMANCE METRICS OF NORMAL ONNX MODELS

Model	FPS Mean	FPS Min	FPS Max	Latency Mean (ms)	Latency Min (ms)	Latency Max (ms)	Frame Count
XceptionNet	140.64	105.17	199.78	7.11	5.01	9.51	152
MobileNet-V2	423.53	124.98	500.27	2.36	2.0	8.0	152
EfficientNet-B0	318.17	199.59	499.32	3.14	2.0	5.01	153
ResNet-18	245.48	166.5	333.41	4.07	3.0	6.01	152

2) Real-Time Performance of Simplified ONNX Models: The same models were also converted to simplified ONNX format using the `onnx-simplifier` library. This simplification reduces computational complexity during inference. Table VIII presents the performance metrics of the simplified ONNX models:

TABLE VIII
REAL-TIME PERFORMANCE METRICS OF SIMPLIFIED ONNX MODELS

Model	FPS Mean	FPS Min	FPS Max	Latency Mean (ms)	Latency Min (ms)	Latency Max (ms)	Frame Count
XceptionNet	139.2	99.89	166.63	7.18	6.0	10.01	150
MobileNet-V2	432.59	124.98	500.10	2.31	2.0	8.0	156
EfficientNet-B0	322.68	249.44	499.86	3.10	2.0	4.01	153
ResNet-18	245.48	166.09	333.36	4.07	3.0	6.02	152

3) Interpretation and Comparison: Comparing the performance of normal ONNX models (Table VII) and simplified ONNX models (Table VIII):

- **XceptionNet:** The average FPS was 140.64 in normal ONNX and 139.2 after simplification. Latency values were very close (7 ms). The difference is minimal.
- **MobileNet-V2:** After simplification, the FPS increased slightly from 423.53 to 432.59. A minor latency improvement was also observed (from 2.36 ms to 2.31 ms).
- **EfficientNet-B0:** The FPS and latency remained almost the same before and after simplification.
- **ResNet-18:** No significant difference was observed between the normal and simplified versions.

Overall, simplified ONNX models provide small improvements in resource usage but do not significantly affect the major FPS and latency metrics. The `onnx-simplifier` step effectively reduces complexity while maintaining performance.

Thanks to these results, it was confirmed that all models are capable of running in real-time scenarios, with high FPS and low latency suitable for practical deployment.

VII. CONCLUSIONS

In this project, the multi-class facial expression recognition problem was addressed using the FER2013 dataset. Due to the clear imbalances in the dataset — particularly the underrepresentation of the *Disgust* class — different sampling strategies (undersampling, oversampling, hybrid sampling) and various loss functions (Cross Entropy, Class Weighted, Focal Loss) were applied to improve model performance.

Additionally, face alignment operations using FaceNet MTCNN were implemented to obtain consistent, normalized facial images instead of raw and irregular data. Fine-tuning was performed on XceptionNet, MobileNet-V3-Large-100, EfficientNet-B0, and ResNet-18 models through a transfer learning approach. Test results and class-based evaluations revealed that the **XceptionNet** model especially demonstrated superior performance.

Models converted into ONNX format achieved high FPS and low latency in real-time applications, indicating the system's suitability for practical usage. Furthermore, using the LIME method, insights into the decision-making processes of the models were obtained, identifying reasons behind misclassifications.

What We Learned and Gained:

- Applying sampling strategies and special loss functions is effective in building a balanced facial expression recognition system that includes all emotion classes.
- Face alignment significantly enhances model performance, particularly for underrepresented classes like *Disgust*.
- Transitioning to ONNX format and applying model simplifications optimizes model performance (FPS, latency) for real-time applications.
- LIME-based explainability analysis provided valuable guidance for understanding model focus areas and future improvements.

Limitations and Challenges:

- The resolution and grayscale nature of the dataset limited the learning of fine-grained features.
- The very low sample count for the *Disgust* class continued to cause confusion in some models.
- Broader testing with more diverse and larger datasets is recommended for further development.

Comparative Model Performance:

- In a similar study using the same dataset, an accuracy score of 0.6485 was reported by applying Transfer Learning + Fine-Tuning on the Xception model [11].
- In contrast, the model developed in this project — using XceptionNet + Cross Entropy Loss + FaceNet MTCNN Alignment — achieved an accuracy score of **0.6676** on the test set.
- This demonstrates that the proposed approach significantly improves the overall accuracy of the model.

Suggestions for Future Work:

Future studies may consider:

- Using high-resolution and multi-channel (RGB) facial images to increase feature extraction capacity.
- Investigating different face alignment algorithms and advanced data augmentation strategies.
- Exploring hardware optimization techniques (e.g., TensorRT integration) for ONNX models to further accelerate real-time inference.

In conclusion, this project provides effective solutions to challenges like data imbalance and face misalignment on the FER2013 dataset. It offers a balanced and high-performance model for real-time facial expression recognition systems, while integrating interpretability and optimization practices to enhance practical deployment potential.

REFERENCES

[1] Utku.Murat.Atasoy, "Kod linki," https://drive.google.com/drive/folders/1Nr4n_tVyyQNQR8y6CTdmh4-xb0j9qJMR?usp=sharing, rapora gömülü kod klasörleri için bağlantı.

[2] Utku..Murat..Atasoy, "Sunum videosu," https://youtu.be/779Kehtc7i8?si=jvS4lshXKJmPJ-Kr, youTube üzerinden sunum.

[3] Y. Khaireddin and Z. Chen, "Facial emotion recognition: State of the art performance on fer2013," https://paperswithcode.com/sota/facial-expression-recognition-on-fer2013, 2021, submitted on 8 May 2021.

[4] G. P. Kusuma and J. A. P. Lim, "Emotion recognition on fer-2013 face images using fine-tuned vgg-16," 2020, submitted on 10 November 2020.

[5] O. C. Oguine, K. A. Kinfu, K. J. Oguin, and D. Ofuani, "Hybrid facial expression recognition (fer2013) model for real-time emotion classification and prediction," 2022, june 2022.

[6] L. Zahara, P. Musa, E. P. Wibowo, I. Karim, and S. B. Musa, "The facial emotion recognition (fer-2013) dataset for prediction system of micro-expressions face using the convolutional neural network (cnn) algorithm based raspberry pi," *IEEE Xplore*, 2020, date Added to IEEE Xplore: 24 December 2020.

[7] S. Qi, X. Zuo, W. Feng, and I. G. Naveen, "Face recognition model based on mtcnn and facenet," *IEEE*, 2022, trained on LFW dataset, 86.0% accuracy.

[8] C. Wu and Y. Zhang, "Mtcnn and facenet based access control system for face detection and recognition," *Automatic Control and Computer Sciences*, vol. 55, no. 1, pp. 102–112, 2021.

[9] M. K. Roy, P. Dwibedi, A. Singh, R. P. Chakraborty, and M. K. H. Mondal, "Mtcnn and facenet-based face detection and recognition model for attendance monitoring," in *Conference Paper*, 2024, first Online: 18 February 2024.

[10] Codejay12, "Facial-expression-recogination," https://github.com/codejay12/Facial-Expression-Recogination, 2020.

[11] L. Rose, "Facial-expression-recognition," https://github.com/leorrose/Facial-Expression-Recognition, 2020.