AFER: Advancements in Facial Expression Recognition.

Cecil Joseph
Artificial Intelligence

February 6, 2024

Abstract

Facial Expression Recognition (FER) holds significant importance in computer vision, serving various fields such as healthcare, human-computer interaction, and emotional analysis. This study aims to elevate FER systems by employing advanced techniques, including fine-tuning and architectural improvements, utilizing deep learning models like YOLO (You Only Look Once).

Drawing upon comprehensive research insights, the investigation integrates diverse methodologies to refine FER systems. It explores approaches that streamline model optimization, adapt to varying environmental conditions, and account for cultural nuances. All of these considerations are examined within the framework of deep learning models, particularly YOLO and Xception.

Furthermore, the study delves into an exploration of the FER2013 database and YOLO-processed FER2013 data. The primary goal is to provide a thorough understanding of dataset characteristics and the necessary preprocessing steps involved in the FER task. This comprehensive approach not only advances FER systems but also contributes to a broader understanding of dataset intricacies and optimization techniques within the domain of facial expression recognition.

Contents

List of Figures

List of Algorithms

List of Tables

1 Introduction

Facial emotion recognition involves the nuanced identification of facial expressions conveying fundamental emotions like fear, happiness, and disgust. This capability holds substantial significance across various applications, including human-computer interactions, digital advertising, online gaming, customer feedback assessment, and healthcare [4, 1]. Despite marked progress in computer vision, achieving high accuracy in emotion recognition from images captured under controlled conditions, the field continues to grapple with challenges when confronted with naturalistic conditions, such as high intra-class variation and low inter-class variation [16].

This study exclusively focuses on investigating and comparing the YOLO implementation and its performance concerning the Xception model, aiming to refine the implementation of the Xception model or explore potential architectural alterations to YOLO models. The focal point is a meticulous examination of the FER 2013 database, emphasizing specific emotions: 'Neutral,' 'Surprise,' 'Sad,' 'Happy,' 'Fear,' 'Disgust,' and 'Angry.'

The intricate realm of human emotions conveyed through facial expressions has sparked considerable interest, leading to the development of numerous methodologies and algorithms dedicated to recognizing and interpreting these expressions [5, 24]. Challenges persist in achieving robustness, adaptability, and accuracy within FER systems, particularly concerning diverse environmental conditions, cultural nuances, and subtle expressions.

In response, this research delves deeper into FER intricacies by exploring and harnessing the potential of YOLO and Xception models. YOLO, renowned for its object detection capabilities, and Xception, recognized for its depthwise separable convolutional layers, present promising avenues for refining FER systems [29, 11].

This ambitious pursuit marks a departure from conventional practices within Facial Expression Recognition (FER), meticulously examining the fine-tuning of the Xception model and contemplating substantial architectural adaptations within the YOLO framework. The primary objective is to elevate these models, granting them seamless adaptability that transcends diverse and challenging environmental conditions.

This exploration signifies a comprehensive journey, intricately dissecting the fundamental principles of YOLO and Xception, representing an immersive quest to unveil their inherent capabilities. The renowned FER2013 Kaggle Challenge dataset, comprising 48x48 pixel grayscale images portraying faces, serves as the foundation for our investigation. These images depict faces that have been automatically aligned for a more or less centered position, ensuring uniform spatial occupation. However, the dataset presents challenges with empty pictures and instances of misclassification, adding complexity to its utilization.

Throughout the ensuing chapters, this study narrates an immersive odyssey characterized by methodical analyses, intricate experiments, and profound discoveries. Each stride in this exploration holds the potential not only to advance FER systems but also to reshape our comprehension and drive the evolution of Facial Expression Recognition into unprecedented realms of excellence.

2 Background

In the realm of recognizing facial emotions, the predominant focus in recent research papers revolves around deep learning methodologies, particularly centering on Convolutional Neural Networks (CNNs). The subsequent section aims to establish the fundamentals of CNNs.

2.1 Convolution Neural Network

Convolutional Neural Networks (CNNs) are specialized neural networks specifically designed to handle data organized in a grid-like topology (see Figure 1).

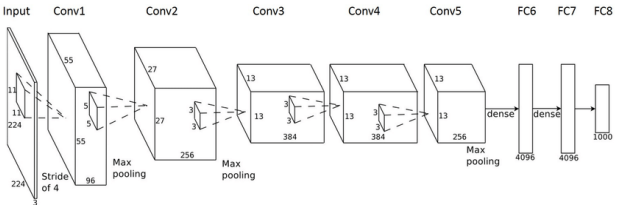


Figure 1: Typical CNN Architecture

In conventional approaches, considerable emphasis was placed on selecting filters (e.g., Gabor filters) and designing architectures to extract the utmost information from images. Nevertheless, the advent of deep learning and enhanced computational capacities has led to increased automation of this process. CNNs are named after the convolution of an initial image input with a set of filters. Essential parameters encompass the number of filters, their dimensions, and the stride length, typically falling within the range of 2 to 5, indicating the step size for filter convolution across the image.

The convolutional process generates an output in three dimensions, deviating from the original two-dimensional image. The interpretation of filters, particularly when their number is substantial, poses a challenge for human understanding, as each filter serves specific purposes such as detecting curves, edges, or textures. Following the creation of this volumetric output, it undergoes flattening and is subsequently fed into a dense neural network.

In each convolutional step, an activation function (typically ReLU) is applied to every input, augmenting the dimensionality of the initial image input. To counteract this increase in dimensionality, a pooling step is subsequently introduced. Pooling entails downsampling features, reducing the number of parameters requiring learning during training. Max-pooling stands out as the most prevalent form of pooling, where, within each dimension of the input image, a max-pooling operation is executed over a specified height and width (typically 2x2), selecting the maximum value among the four pixels. This methodology is grounded in the intuition that the maximum value holds greater significance when classifying an image.

A convolutional neural network comprises essential components, including convolution layers, activation layers (where activation functions are applied), pooling layers, and fully connected layers, resembling a dense neural network.The configuration of these layers is subject to modification. For example:

$$\text{ReLU}\left(\text{MaxPool}\left(\text{Conv}(X)\right)\right) = \text{MaxPool}\left(\text{ReLU}\left(\text{Conv}(X)\right)\right)$$

This equation demonstrates the flexibility to interchange the order of operations, such as ReLU, MaxPool, and Convolution, within the network architecture.

In the field of image classification, it is common to integrate multiple layers of convolution and pooling. This strategy facilitates the modeling of intricate structures within the data. The central emphasis on refining models in deep learning often centers around determining the optimal architecture [3]. Remarkably, specific well-known algorithms developed by entities such as Microsoft or Google extend to depths surpassing 150 hidden layers.

2.2 XCeption architecture and convolutions using Depthwise Separable layers

Xception embodies an advanced convolutional neural network architecture that integrates Depthwise Separable Convolutions [12]. Pioneered by Google researchers, this design reimagines Inception modules as an intermediary step between conventional convolution and the depthwise separable convolution operation, involving a depthwise convolution followed by a pointwise convolution. Treating a depthwise separable convolution akin to an Inception module with a multitude of towers inspired the formulation of a novel deep convolutional neural network architecture influenced by Inception. In this updated structure, Inception modules are substituted with depthwise separable convolutions. The data undergoes an initial entry flow, followed by eight iterations of a middle flow, and ultimately through the exit flow. It is noteworthy that all Convolution and SeparableConvolution layers are followed by batch normalization.

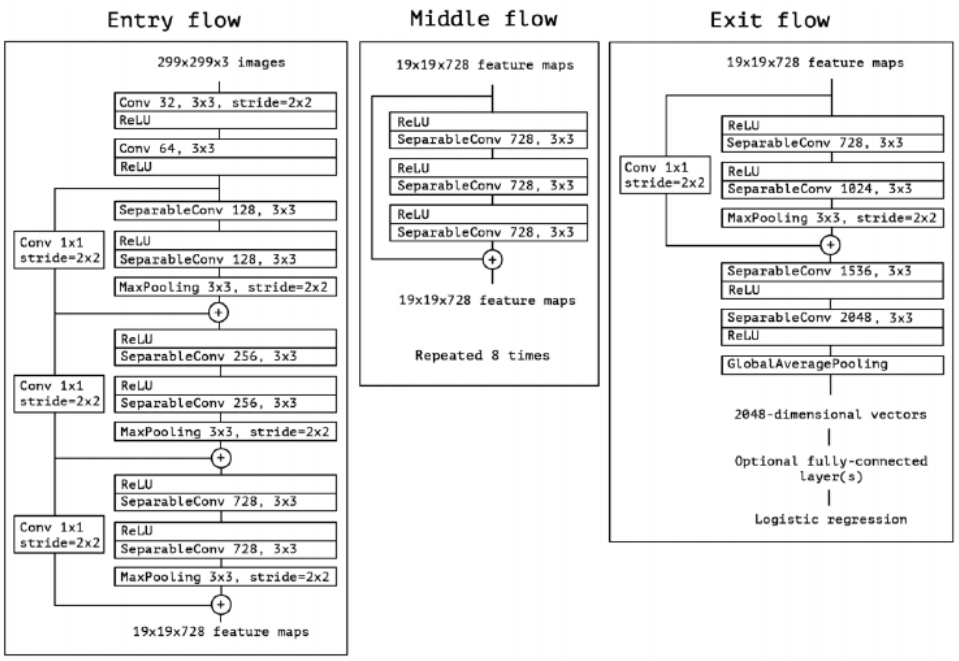


Figure 2: Xception Structure

The Xception architecture, as highlighted in the Figure 2, the work by Chollet (2017), has consistently outperformed VGG-16, ResNet, and Inception V3 across various classical classification challenges. It stands out as an efficient design grounded in two key principles: Depthwise Separable Convolution and the incorporation of shortcuts between convolution blocks, reminiscent of ResNet. The use of Depthwise Separable Convolutions provides a noteworthy departure from traditional convolutions, offering significantly enhanced efficiency in terms of computation time. To comprehend the rationale behind this improvement, let's delve into the nature of convolution operations. Convolution operations are inherently computationally intensive, and recognizing this, there is a motivation to explore more efficient alternatives.

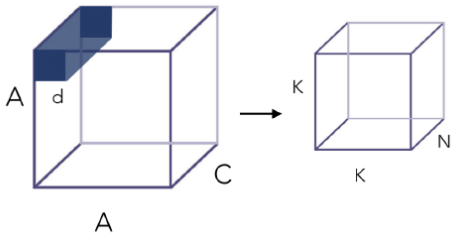


Figure 3: Convolutions

The input image has a certain number of channels C, say 3 for a color image. It also has a certain dimension A, say 100×100. We apply on it a convolution filter of size $d \times d$, say 3×3.

For 1 Kernel, we have a substantial number of operations:

$$K^2 \times d^2 \times C$$

where K is the resulting dimension after convolution, which depends on the padding applied (e.g., padding *same* would mean $A = K$).

Therefore, for N Kernels (depth of the convolution):

$$K^2 \times d^2 \times C \times N$$

To overcome the cost of such operations, depthwise separable convolutions have been introduced. They are themselves divided into 2 main steps:

1. Depthwise Convolution
2. Pointwise Convolution

Depthwise convolutions Initially, the Depthwise Convolution serves as the primary step, wherein, instead of employing a convolution of size $d \times d \times C$, we opt for a convolution of size $d \times d \times 1$. In essence, the computation of the convolution is performed individually for each channel, one at a time.Depthwise convolutions have found widespread adoption in various neural network architectures for efficient computation. While the concept is prevalent, specific adaptations of depthwise convolutions may vary across different works. For instance, the introduction of depthwise separable convolutions in the context of mobile architectures can be attributed to the MobileNet paper by Howard et al. (2017) [20]. An illustration of the Depthwise Convolution process is presented below:

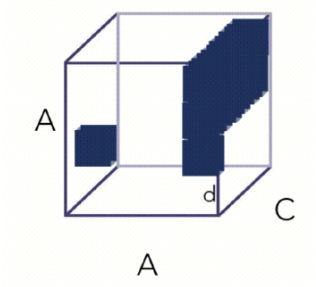


Figure 4: Illustration of the Depthwise Convolution Process

As a result, a preliminary volume is generated with dimensions $K \times K \times C$, rather than $K \times K \times N$ as in the previous case. Up to this point, the convolution operation has been performed for a single kernel per filter of the convolution, not for all N of them. This brings us to our second step.

Pointwise convolutions The Pointwise convolution executes a conventional convolution with a size of $1 \times 1 \times N$ over the $K \times K \times C$ volume. This results in the creation of a volume with dimensions $K \times K \times N$, as seen previously. This methodology effectively reduces the overall number of operations in comparison to regular convolutions by a factor proportional to $1/N$.

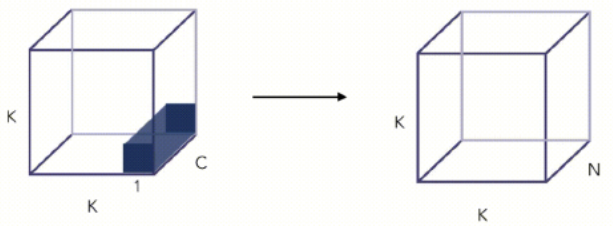


Figure 5: Illustration of the Pointwise Convolution Process

2.3 YOLO (You Only Look Once)

The proliferation of FER (Facial Emotion Recognition) technology can be primarily attributed to recent advancements in pattern recognition, machine learning, and biometric analysis, along with the increased utilization of cameras [9]. YOLO (You Only Look Once) stands out as a popular choice for applications requiring swift and accurate object recognition, such as surveillance, self-driving cars, and gaming [9].

Incorporating emotion recognition into the YOLO architecture has been a recent development, enabling the categorization of people's emotional states in images or videos [25]. This entails training the YOLO model on substantial datasets like Affect-Net, comprising faces and associated emotions, to forecast emotions—commonly identifying happiness, sadness, anger, surprise, contempt, among others [25].

YOLO-based emotion detection has exhibited speed and accuracy, finding application in diverse fields like security, human-computer interaction, and market research [30]. Cameras play a pivotal role in capturing real-time human responses for facial expression research, utilizing the distinctive flex and contraction of facial muscles, facilitating deep learning algorithms in emotion recognition [7].

Variables impacting the efficacy of YOLO-based emotion detection systems include training data quality, model size, and the diversity of facial expressions depicted in images [7]. Basic human emotions—surprise, contempt, rage, fear, happiness, and sadness—are recognizable through a spectrum of facial expressions, encompassing mouth position, eye and brow movement [2].

The versatility of the YOLO algorithm enables its application in diverse contexts, such as gauging people's interest in interaction or employing face recognition for surveillance and security authentication purposes [15]. Advancements in pattern recognition and computer vision, especially in automated face detection, have seen considerable strides due to deep learning techniques, distinguishing YOLO from earlier architectures like R-CNN [15].

The YOLO (You Only Look Once) series comprises several versions, each contributing advancements to real-time object detection. YOLO v1, proposed by Redmon and Divvala in 2016, revolutionized object detection by dividing images into a grid and predicting bounding boxes and class probabilities for each grid cell [29]. YOLO v2, also known as YOLO9000 (2017), addressed limitations of its predecessor by introducing anchor boxes and the ability to detect a large number of object categories [27]. YOLO v3 (2018) further improved accuracy by introducing a feature pyramid network and leveraging multiple scales for detection, making it versatile across different object sizes [28]. YOLO v4 (2020) built upon these advancements, introducing CSPDarknet53 as the backbone architecture and incorporating techniques like Mish activation, CIOU loss, and SAM block for enhanced performance [8].

YOLOv5, the fifth iteration of the object detection system, introduced PyTorch, enabling real-time object detection for computer vision tasks. Developed by Ultralytics LLC, its grid-based approach uses a deep neural network to predict object presence in grid cells. The modular and scalable architecture, featuring CSP ResNet or MobileNetV3, allows easy customization. YOLOv5 achieves efficiency and speed, performing multiclass object detection with bounding box predictions and class probabilities.

YOLOv6, an advanced addition to the YOLO (You Only Look Once) algorithm series, is designed for quick and effective real-time object identification. Its architecture includes a deep convolutional neural network with a ResNet-inspired backbone for feature extraction. The neck, composed of convolutional layers, refines these features. The fully connected head layer predicts bounding boxes and class probabilities, incorporating anchor boxes for alignment with ground truth. YOLOv6 employs a loss function with localization, objectness, and classification components. To enhance accuracy, the algorithm utilizes Non-Maximum Suppression (NMS) for eliminating overlapping and redundant detections.

YOLOv7, the seventh version of the You Only Look Once (YOLO) series, is a real-time object detection system built on a single convolutional neural network (CNN) architecture. Designed for computational efficiency, it balances speed and accuracy. The YOLOv7 architecture consists of a backbone network, employing ResNet for feature extraction from input images. The neck network refines extracted features through convolutional and pooling layers. The head network analyzes refined feature maps for object detection, predicting bounding boxes using anchor boxes. YOLOv7 handles various object shapes by employing anchor boxes with different aspect ratios. The architecture utilizes a multi-task loss function to minimize variations in class probabilities, confidence intervals, and bounding boxes between predictions and real-world data.

YOLOv8 stands as the most recent iteration of YOLO developed by Ultralytics. Representing a cutting-edge and state-of-the-art (SOTA) model, YOLOv8 extends upon the accomplishments of its predecessors by introducing novel features and enhancements, aiming for improved performance, flexibility, and efficiency. This version supports a comprehensive spectrum of vision AI tasks, encompassing detection, segmentation, pose estimation, tracking, and classification. Such versatility empowers users to harness the capabilities of YOLOv8 across a wide array of applications and domains. We have used a pre-trained YOLO v8 model for emotion detection. The input consists of a 48x48 images and its annotation data through the algorithm 1.

Training the YOLO network for emotion recognition entails datasets labeled with emotions, enabling

the model to identify faces and infer associated emotions, outputting bounding boxes around faces with predicted emotions [10]. Its real-time performance distinguishes YOLO for emotion identification, making it suitable for applications requiring swift and efficient emotion detection [10].

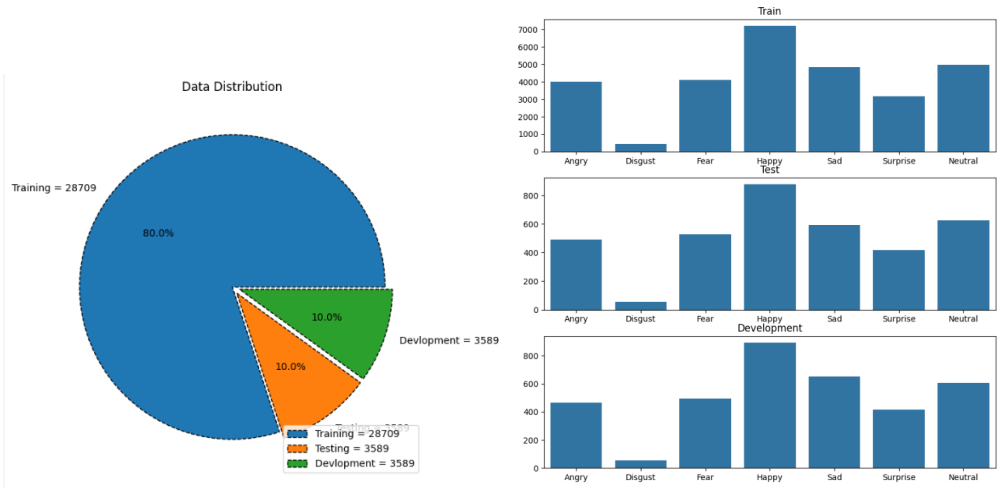
The accuracy of YOLO in identifying emotions hinges on both the training data quality and network complexity [10]. The article concludes by providing a comparative analysis of various YOLO models, offering insights into their respective architectures [9].

3 System Architecture and Construction

In the realm of system architecture and construction, data exploration and visualization serve as foundational pillars, providing crucial insights into the dataset's characteristics and informing subsequent design decisions. Before delving into the intricacies of system construction, it is imperative to embark on an exploratory journey through the data landscape, unraveling patterns, anomalies, and potential challenges that may lie within.

3.1 Data exploration and visualization

First of all, when exploring the data of the FER2013 data set, we observe that there is an imbalance in the number of images by class (emotion), As depicted in the Figure 6b. Establishing a dataset that is balanced, with equal representation of each class, stands as a pivotal initial stage in machine learning. This step is essential to promote impartial model training, as it evens out the instances across diverse classes, preventing the model from leaning heavily on any particular majority group. A balanced dataset enhances overall model performance and the ability to generalize, especially in situations where imbalances among classes might otherwise cause biased predictions. we will discuss balancing its effects in training in later section. The train set has 28709 images, the test set has 3589 images, and Development set has 3589 images



(a) Dataset distribution (b) Emotion distribution

Figure 6: FER-2013 Distribution

as clearely visible from Figure 6a. For each image,the data set contains the grayscale color of 2304 pixels (48x48), as well as the emotion associated.

The challenge is that some pictures are miss-classied,while other images only show part of a face. For example, the two images of Figure 7 seem rather clearly miss-classied.

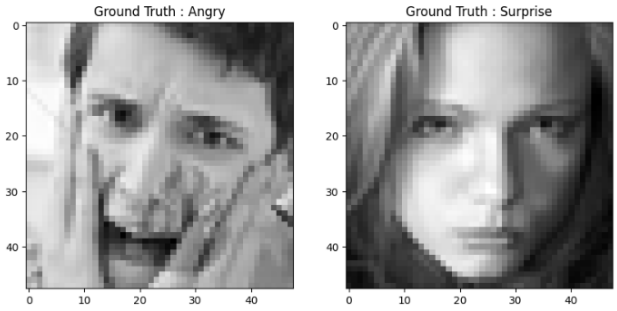


Figure 7: Example of miss-classied images

Examining the significant areas of the image relevant to classification tasks, [14] employs a basic XGBoost classifier on flattened images to assess feature importance. The analysis reveals that crucial regions are predominantly situated near the eyes and around the mouth, as illustrated in Figure 8.

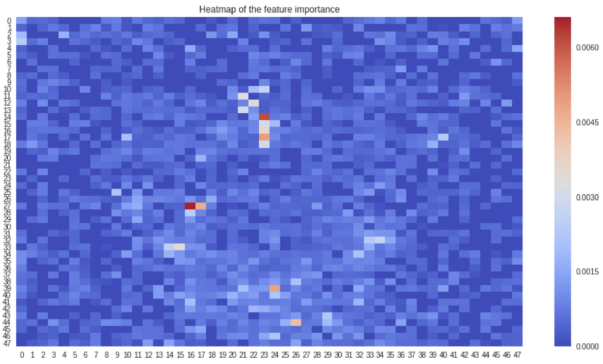


Figure 8: Heatmap of the feature importance

Exploring the average facial expression for each emotion in the dataset provides valuable insights into how emotions are interpreted within the data. A notable observation highlights distinct variations in eye axis between anger and happiness, as depicted in Figure 9.

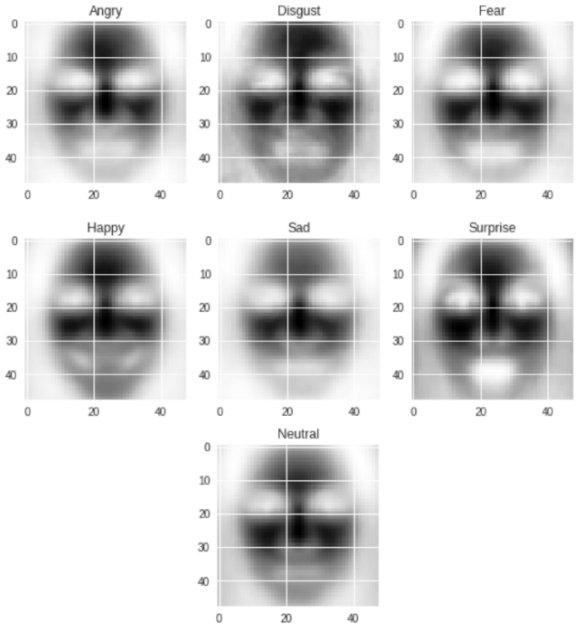


Figure 9: Average Face per emotions

3.1.1 HOG and Landmarks Feature Extraction

In our approach to feature extraction, we employ a comprehensive strategy to enhance the discriminative power and precision of our model. Our focus revolves around three primary feature extraction techniques: HOG sliding windows, HOG features, and facial landmarks, each contributing uniquely to the characterization of input images.

The initial method involves the generation of HOG sliding windows, where the image is systematically partitioned into windows. Within each window, Histogram of Oriented Gradients (HOG) features, as outlined by Dalal and Triggs (2005) [13], are extracted. By sliding the window across the image, local variations in gradients are captured, effectively enabling the model to discern spatial patterns. This proves particularly advantageous for detecting objects with varying shapes and textures.

The second technique revolves around extracting HOG features directly from the entire image, as demonstrated in Figure 3.1.1. This process entails calculating gradients and orientations of pixels, organizing them into histograms within predefined cells, and normalizing these histograms. The resulting HOG features encapsulate crucial information about the image's texture and shape, offering a comprehensive representation of its distinctive features, following the method outlined by Dalal and Triggs (2005) [13].

The third facet of our feature extraction involves the identification of facial landmarks. Utilizing the dlib library, we employ a shape predictor trained on 68 facial landmarks, as discussed by King (2009) [22]. These landmarks pinpoint key locations on the face, providing detailed spatial information about facial structures such as eyes, nose, and mouth. This not only aids in recognizing facial expressions but also contributes valuable geometric details for improved feature representation.

By integrating these diverse feature extraction methods, we construct a rich feature set that captures both local and global information from the input images. The combination of HOG sliding windows, HOG features, and facial landmarks is designed to comprehensively represent the inherent complexities of facial expressions and structures. We systematically evaluate the impact of different feature combinations on training and prediction accuracy, aiming to optimize the performance of our model in diverse scenarios.

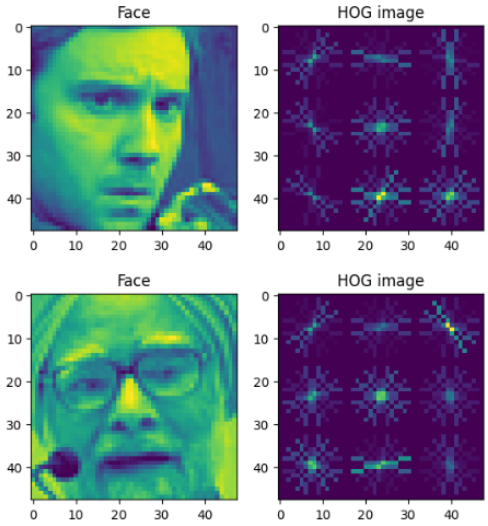


Figure 10: HOG Features Extracted

3.1.2 Visualization of Data in Lower Dimention

When addressing the visualization of data in reduced dimensions, methods like dimensionality reduction become pivotal. These techniques assist in portraying high-dimensional data in a more accessible format, facilitating exploration and interpretation. Among the frequently employed approaches for dimensionality reduction are Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE).

Auto-Encoding for Dimension Reduction The initial step involved the application of auto-encoding to compress the images and achieve dimension reduction. The implemented auto-encoder aims to reduce the dimensions by over 95

This reduction in dimensionality serves multiple purposes, including mitigating the curse of dimensionality and enhancing computational efficiency. By transforming the original high-dimensional data into a more compact representation, we not only reduce the computational load but also potentially enhance the model's ability to capture meaningful patterns and features.

Furthermore, the choice of a 12 * 12 pixel dimension for the encoded image strikes a balance between maintaining essential information and achieving significant compression. This compression facilitates a streamlined representation of the input data, enabling subsequent analysis and processing with a reduced computational overhead.

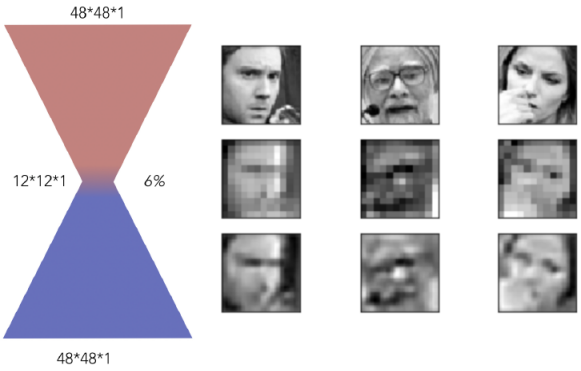


Figure 11: Auto-Encoding the input images

Through the encoding, we lose a lot of information [19, 31]. This might be problematic since we do not manage to fully understand the main features, although the restriction of 12 * 12 might seem a bit ambitious [6].

Since auto-encoding allows dimension reduction, we can again reduce the dimension to a dimension of 2 in order to be able to represent the different classes on a simple graph using t-Stochastic Neighbor Embedding (t-SNE) or Principal Component Analysis (PCA) techniques [23].

The use of t-SNE or PCA after auto-encoding enables the exploration and visualization of complex data structures in a more interpretable and visually accessible manner [17].

Principal Component Analysis on FER-2013 Principal Component Analysis (PCA), a linear dimensionality reduction technique, is adept at capturing the maximum variance within data through its principal components [21]. This method transforms the original features into a fresh set of uncorrelated variables, with the initial principal components often capturing the majority of the variance. This transformation facilitates the visualization of data in a reduced-dimensional space, providing a concise representation that simplifies the exploration and interpretation of intricate datasets.

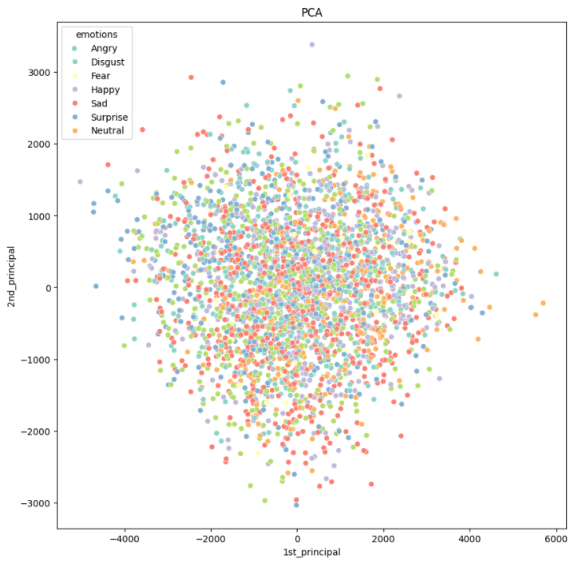


Figure 12: PCA on FER-2013

In the case of PCA, the algorithm identifies directions (principal components) along which the data varies the most. Notably, it does not consider class labels or groupings during the dimensionality reduction process. Consequently, if the underlying data structure involves complex patterns or non-linear relationships, PCA might not reveal clear separations between different groups, as illustrated in Figure 12.

In situations like FER-2013, where the data exhibits inherent clusters or groups not well-captured by linear combinations of the original features, non-linear dimensionality reduction techniques such as t-Distributed Stochastic Neighbor Embedding (t-SNE) might be more suitable. T-SNE focuses on preserving local similarities and can reveal more intricate structures in the data. Therefore, we opted for exploration using t-SNE.

Distributed Stochastic Neighbor Embedding (t-SNE) on FER-2013 The subsequent step involved exploring FER-2013 using t-Distributed Stochastic Neighbor Embedding (t-SNE), a non-linear dimensionality reduction method [26]. t-SNE is designed to preserve pairwise similarities among data points, excelling in visualizing clusters within high-dimensional datasets. It emphasizes local structures and proves particularly adept at unveiling intricate patterns in complex datasets.

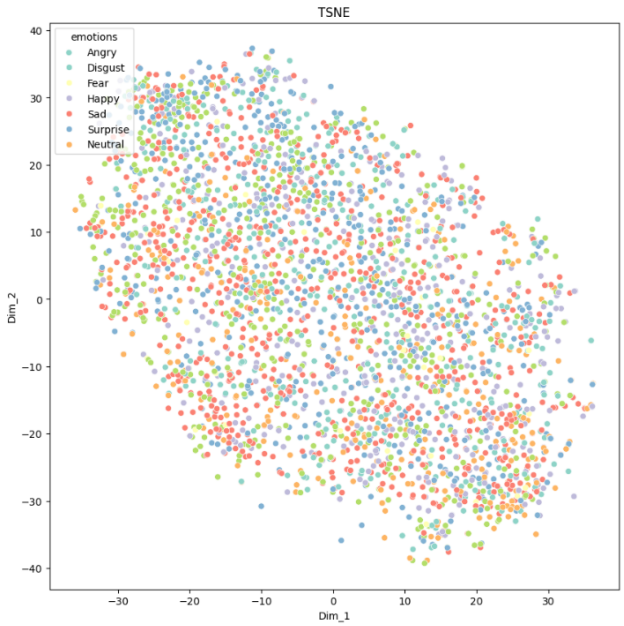


Figure 13: Distributed Stochastic Neighbor Embedding (t-SNE) on FER-2013

In Figure 13 t-SNE, like PCA, might not inherently reveal distinct clusters or separations between different groups in the data. Despite being a non-linear dimensionality reduction technique that emphasizes preserving pairwise similarities and capturing local structures, t-SNE does not guarantee well-separated clusters in the reduced-dimensional space.

Both t-SNE and PCA aim to represent high-dimensional data in a lower-dimensional form, but their approaches differ. t-SNE is known for its effectiveness in revealing local patterns and capturing complex structures, but the global structure might not always be well-defined, leading to data points scattered throughout the reduced space without clear boundaries between groups and we can see this throughout in the Figure 13.

3.1.3 Dynamic Annotation: Leveraging Superset Solutions for Subset Labeling, FER-2013 Dataset Yolo Processed.

In the context of working with the 48x48 images from the FER-2013 dataset, the task at hand involves preparing an annotated dataset suitable for YOLO (You Only Look Once), a popular object detection algorithm. The FER-2013 dataset inherently categorizes images based on emotions, providing a foundation for annotation. However, the challenge arises in the annotation process itself.

To address the annotation hurdle, we developed a systematic approach leveraging automated techniques. Our plan involves employing the dlib.get_frontal_face_detector() function, which is a part of the dlib library, renowned for its facial detection capabilities. This function enables us to automatically detect faces within the images. Subsequently, we use the identified face coordinates to annotate the images accordingly.

By integrating this automated face detection and annotation process described in the Algorithm 1, we aim to streamline the preparation of a YOLO-ready dataset from the FER-2013 dataset, making it conducive for training an object detection model that can recognize and localize facial expressions. This approach not only simplifies the annotation task but also enhances the efficiency of dataset preparation for subsequent machine learning tasks.

Algorithm 1 Yolo Annotation Procedure

function ANNOTATE(trump_face, label)
  if img is None **then**
    print("Image not loaded. Check the file path or file format.")
  end if
  face_detect ← dlib.get_frontal_face_detector()
  coordinates ← face_detect(img, 1)
  face_rect ← coordinates[0]
  $x1$ ← face_rect.left()
  $y1$ ← face_rect.top()
  $x2$ ← face_rect.right()
  $y2$ ← face_rect.bottom()
  normalized_x_center ← $(x1 + x2)/(2 \times \text{width})$
  normalized_y_center ← $(y1 + y2)/(2 \times \text{height})$
  normalized_width ← $(x2 - x1)/\text{width}$
  normalized_height ← $(y2 - y1)/\text{height}$
  yolo_annotation ← label, normalized_x_center, normalized_y_center, normalized_width, normalized_height
  return yolo_annotation, $(x1, y1)$, $(x2, y2)$
end function

Through the utilization of this methodology, we ensure that annotations are exclusively generated for images in which successful face detection has taken place, as illustrated in Figure 14. This step is crucial as a preprocessing measure, and its adoption enables us to implicitly bypass the necessity for explicitly executing such preprocessing steps for images lacking successful face detection. Essentially, this approach streamlines the annotation generation process by concentrating solely on images with detected faces, thereby enhancing the overall workflow's efficiency.

In addition, this procedure allows us to develop an approach that benefits the labeling process. By first identifying a superset problem and employing the superset solution, we can annotate the subset problem. For example, applying this strategy to facial recognition, we address the broader challenge of recognizing faces in images, and then leverage this solution to annotate specific facial expressions. Similarly, in the context of identifying mushrooms, we address the overarching challenge of recognizing mushrooms in images, and subsequently, use this solution to annotate specific mushroom types. This approach optimizes the labeling procedure by addressing broader categories first and then leveraging those solutions for more specific subsets.

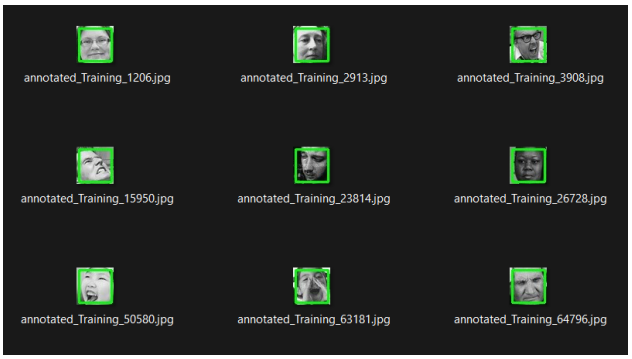


Figure 14: Yolo processed Dataset with Annotation

3.2 XCeption architecture

XCeption presents an innovative architecture composed of Depthwise Separable Convolution blocks and Maxpooling, interconnected through shortcuts reminiscent of ResNet implementations [18]. Notably, in XCeption, the traditional order of Depthwise Convolution followed by Pointwise Convolution is reversed, as exemplified in this instance. This inversion enhances the model's ability to capture intricate patterns and features within the data [14].

The XCeption model [11] is designed not only for superior performance but also for efficient resource utilization. It facilitates quicker training times on GPUs, making it well-suited for large-scale datasets and computationally demanding tasks. Moreover, the architecture enhances real-time prediction by increasing image processing per second, a crucial factor in applications requiring rapid decision-making, such as real-time image analysis or video processing.

One notable advantage of the XCeption architecture lies in its inherent regularization properties, which contribute to mitigating the risk of overfitting. The interconnected blocks and shortcut connections enable the model to learn representations that generalize well to unseen data, making XCeption a robust choice for diverse applications.

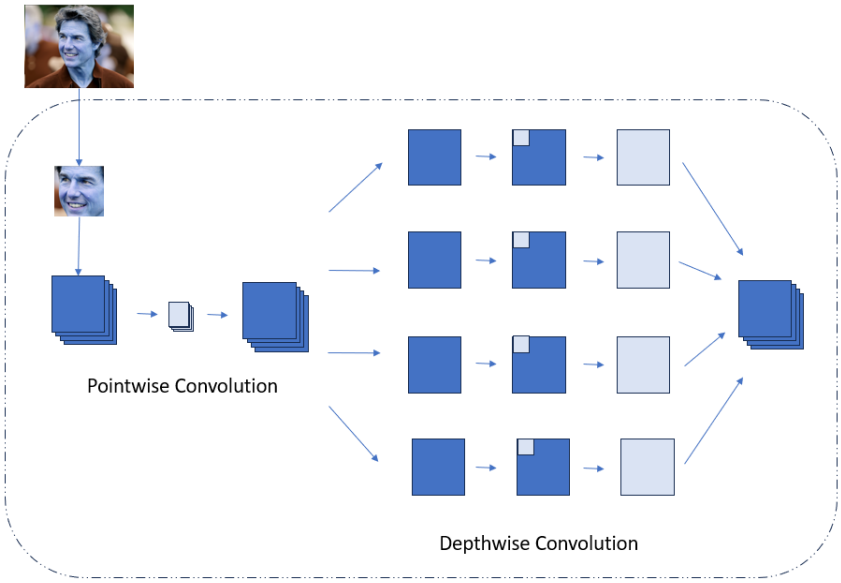


Figure 15: Xception Architecture

3.3 Avoiding Overfitting in Neural Networks - Measures Taken in Implementation

In the realm of neural networks, it is crucial to employ strategies that deter overfitting, ensuring the model generalizes well to new data and avoids memorizing the training set.

3.3.1 Overfitting

Overfitting becomes visually apparent when the training accuracy continues to rise, yet the validation or test accuracy plateaus or fails to show further improvement, In Figure 16 we can see the model getting overfitted.

Relying solely on training accuracy might lead us to choose a model based on its high training accuracy, but this approach is risky because validation accuracy should serve as our benchmark metric. One of the primary hurdles in deep learning is the imperative task of mitigating overfitting.

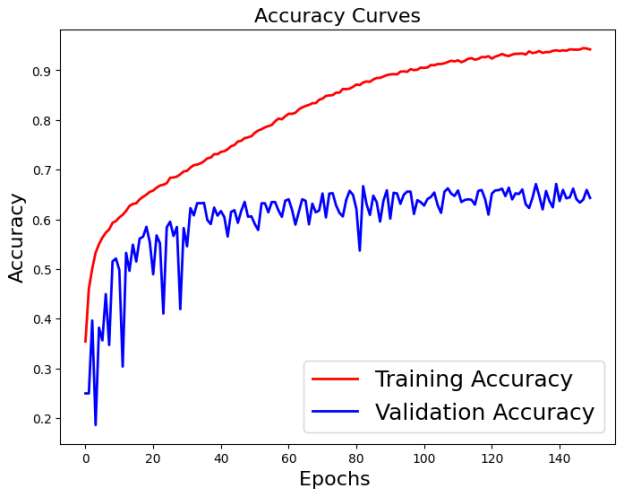


Figure 16: Overfitting in Xception model Observed

3.3.2 Ensuring a Balanced Dataset for Enhanced Model Accuracy

A crucial step in optimizing machine learning models for accuracy involves establishing a well-balanced dataset across classes. Achieving equal representation for each class is foundational in mitigating biases and fostering impartial model training. By evenly distributing instances across diverse classes, the model is prevented from favoring any particular majority group, thereby enhancing its overall performance and generalization capabilities. This is particularly important in scenarios where imbalances among classes could lead to biased predictions.

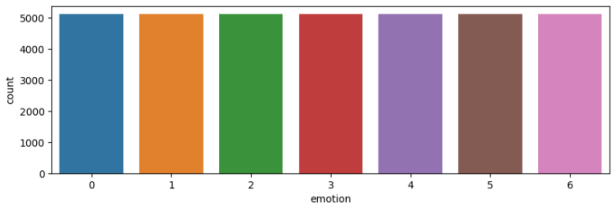


Figure 17: Balanced FER-2013 data subset

To address class imbalance challenges, an additional task involved the utilization of oversampling. This technique involves duplicating instances from each class to align with the calculated average count. The goal is to alleviate imbalances and create a more equitable representation of different classes within the dataset. The oversampled data from each class is consolidated into a unified DataFrame, promoting fairness in model training.

As depicted in Figure 17, a balanced dataset was utilized in the model development. This approach is versatile and applicable across various domains in machine learning. It serves as an effective strategy to improve model training outcomes by ensuring a balanced distribution of instances, thereby contributing to unbiased and more robust predictions. As we can see in the Table 2 there is significant accuracy improvement in the balanced Dataset.

3.3.3 Data Augmentation

Data augmentation serves as a widely adopted strategy in image classification to counteract overfitting. The idea is to apply minor transformations, such as shifts or scaling, to the input images, artificially augmenting the dataset. in keras we can do the below actions to obtain Data Augmentation.

The typical representation of the transformations would be as follows:

```
datagen = ImageDataGenerator(
        zoom_range=0.2,% randomly zoom into images
        rotation_range=10,% randomly rotate(degrees, 0 to 180)
        width_shift_range=0.1,% randomly shift images horizontally
        height_shift_range=0.1,% randomly shift images vertically
        horizontal_flip=True,% randomly flip images
        vertical_flip=False)% randomly flip images
```

Figure 18: Data Augmentation Code Example

Figure 19: Data Augmentation Example

3.3.4 Early stopping

Early Stopping is a technique used to halt the training process when a specific criterion remains unchanged over consecutive epochs. For instance, if the objective is to see an improvement in validation accuracy and the algorithm is set to cease training if no improvement occurs over a span of 20 epochs.

When training your model, the code with both Data Augmentation and Early Stopping would typically appear as shown in the Figure 20.

```
earlystopping = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
# Save the best model with lower validation loss
checkpointer = ModelCheckpoint(filepath="Emotion_weights_Xception_Std.hdf5", verbose=1, save_best_only=True)
history = xception.fit(
    datagen.flow(X_train, train_labels_one_hot, batch_size=batch_size),
    steps_per_epoch=int(np.ceil(X_train.shape[0] / float(batch_size))),
    epochs=epochs,
    validation_data=(X_test, test_labels_one_hot),
    callbacks=[checkpointer, earlystopping]
)
```

Figure 20: Data Augmentation + Early Stopping Applied during Training

4 Outcomes

As the Approach was to explore the FER2013 Dataset and try to improve the overall accuracy and adapt to the yolo architecture we will discuss the Performance evaluations in 2 sections One which is based on various feature sets and other one specifically for YOLO architecture.

4.1 Evaluation Results for Various Feature Sets and Models

In the quest for optimal model performance, we explored diverse feature sets and architectures, each contributing to distinctive accuracy outcomes as depicted in the table 1.

Histogram of Oriented Gradients (HoG) features, coupled with Support Vector Machine (SVM) classification, yielded an accuracy of 29.20%. Employing facial landmarks as input for an SVM model resulted in an accuracy of 47.5%. A combination of facial landmarks and HoG features for SVM classification improved accuracy slightly to 48.33%. The introduction of a sliding window approach to this combination further enhanced accuracy to 50.5%.

Transitioning to image data, models trained on unbalanced datasets (48x48 pixels) using Resnet18, Mini-Xception, and Xception architectures achieved accuracies of 62.00%, 62.6%, and 69.04%, respectively. The pivotal shift towards dataset balancing significantly improved model performance. In the balanced dataset scenario, Resnet18 achieved an accuracy of 84.84%, Mini-Xception reached 88.24%, and Xception excelled with the highest accuracy at 92.03%.

These results underscore the critical role of dataset balance in enhancing model accuracy. The achievement of 92.03% accuracy with a balanced dataset and the Xception architecture stands out as a noteworthy outcome in our pursuit of model optimization. The exploration of various feature sets and architectures not only provides insights into their individual impacts but also reinforces the importance of considering dataset balance for robust and reliable machine learning models.

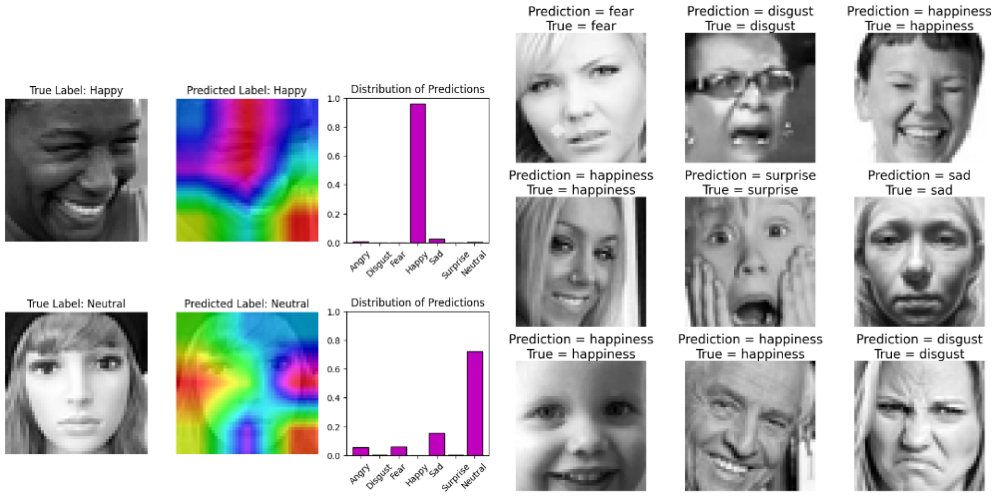
Table 1: Performance Results for Different Feature Sets and Models

Features	**Accuracy**
HoG features (SVM)	29.20%
Face landmarks (SVM)	47.5%
Face landmarks + HoG (SVM)	48.33%
Face landmarks + HoG on sliding window (SVM)	50.5%
Image Data (48x48 pixel) Unbalanced (Resnet18)	62.00%
Image Data (48x48 pixel) Unbalanced (Mini-Xception)	62.6%
Image Data (48x48 pixel) Unbalanced (Xception)	69.04%
Image Data (48x48 pixel) Balanced (Resnet18)	84.84%
Image Data (48x48 pixel) Balanced (Mini-Xception)	88.24%
Image Data (48x48 pixel) Balanced (Xception)	92.03%

Table 2: Impact of Dataset Balancing on Accuracy

Model	**Unbalanced Accuracy**	**Balanced Accuracy**
Resnet18	62.00%	84.84%
Mini-Xception	62.6%	88.24%
Xception	69.04%	92.03%

Balancing the dataset significantly improved the accuracy of the models, as demonstrated in the table 2 above. The models trained on a balanced dataset (Image Data Balanced) consistently outperformed their counterparts trained on an unbalanced dataset (Image Data Unbalanced). The enhancement in accuracy underscores the importance of dataset balancing in improving model performance.



(a) Activation Map (b) Xception Prediction

Figure 21: Xception Model Predictions and Activations

Activation maps, crucial in neural networks like CNNs, visually highlight input regions influencing model predictions. Analyzing these maps, as seen in Figure 21, reveals the model's focus, often concentrating around features like the mouth and eye regions. This insight aids interpretation and refinement, enhancing the transparency and effectiveness of deep learning models.

Saliency maps are visualization techniques used in the field of computer vision and deep learning to understand and interpret the decision-making process of neural networks, particularly in image classification tasks. These maps highlight regions of an input image that have the most significant influence on the model's prediction. By examining these saliency maps, one can gain insights into which parts of the input are crucial for the model's decision.

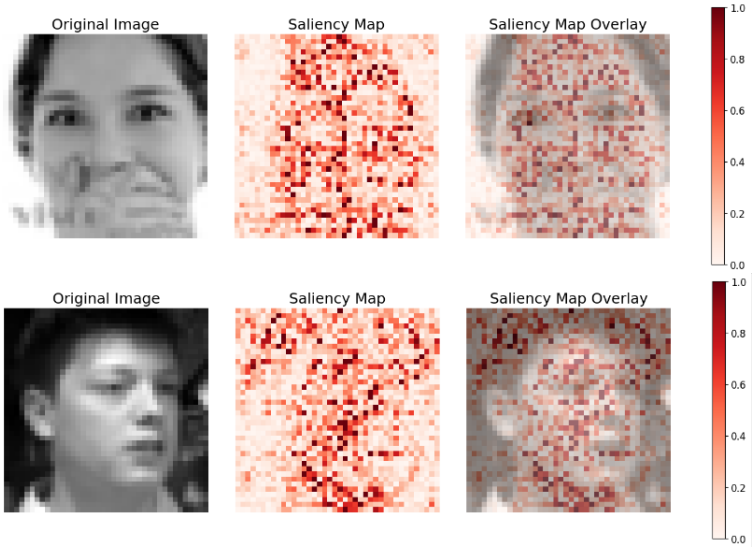


Figure 22: Saliency Map

The saliency maps produced by the pretrained Xception model reveal that the model effectively emphasizes pertinent regions, evident in the clear identification of significant hotspot areas.

4.2 Evaluation Results YoloV8

The table 3 below presents the evaluation results of our model on different emotion classes. The metrics include Precision (P), Recall (R), and Mean Average Precision (mAP) computed at various IoU (Intersection over Union) thresholds.

Table 3: Evaluation Metrics for Emotion Recognition Model -Yolov8

Class	Images	Instances	Box P	R	mAP50	mAP50-95
All	7178	7178	0.647	0.703	0.727	0.727
Anger	7178	958	0.635	0.615	0.694	0.694
Disgust	7178	111	0.574	0.595	0.625	0.625
Fear	7178	1024	0.534	0.537	0.581	0.581
Happy	7178	1774	0.88	0.9	0.952	0.952
Neutral	7178	1233	0.601	0.757	0.729	0.729
Sad	7178	1247	0.572	0.642	0.634	0.634
Surprise	7178	831	0.731	0.877	0.872	0.872

The model demonstrates an overall accuracy rate of 64.7%, as reflected by the Precision (Box P). When examining precision values for specific emotion classes, notable variations emerge. For instance, the "Happy" emotion class exhibits the highest precision at 88%, indicating a strong capability of the model to accurately identify instances of happiness. Conversely, the "Fear" emotion class registers the lowest precision at 53.4%, suggesting a comparatively lower accuracy in recognizing instances of fear.

Furthermore, the recall values across emotion classes exhibit distinct patterns. The "Happy" emotion class achieves a recall of 90%, indicating the model's effectiveness in capturing a substantial proportion of actual positive instances for this emotion. On the other hand, the "Fear" emotion class displays a recall of 53.7%, indicating that the model may overlook a notable portion of actual positive instances related to fear.

In addition to precision and recall, Mean Average Precision (mAP) values are provided at different Intersection over Union (IoU) thresholds, specifically at 50% (mAP50) and 50-95% (mAP50-95). These metrics offer nuanced insights into the model's performance across varying evaluation conditions, providing a comprehensive understanding of its strengths and areas for potential improvement.

4.3 Webcam Implementation

Figure 23 outlines the described process of Model Deployment on a webcam stream. Image processing is executed using OpenCV in Python, with face detection employing a Cascade Classifier. This classifier operates by concatenating multiple classifiers, utilizing information from the previous classifier as supplementary input for the subsequent one in the cascade. Typically applied to regions of interest in an image, the classifier yields a binary result (0 or 1) indicating the likelihood of the region displaying the tested object. Commonly used classifiers include Adaboost, Real Adaboost, Gentle Adaboost, and Logitboost. OpenCV provides pre-trained cascade models specifically designed for face detection.

The described webcam facial emotion algorithm can be exemplified through a brief demonstration. While the classification demonstrates effective performance in practical scenarios, there are still various avenues for improvement.

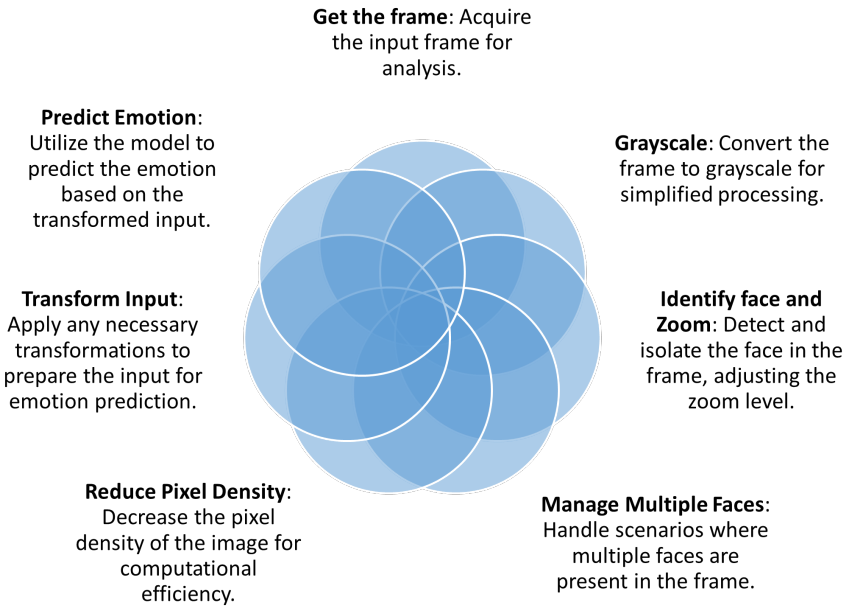


Figure 23: Webcam Model Deployment

Our model performs effectively, as shown in Figure 24 achieving an accuracy rate of 92% with Xception model and with YOLOv8 we are getting a mean average precision of 72%. This can be exemplified through a specific instance to showcase the algorithm's efficiency. Furthermore, efforts have been invested to enable real-time emotion recognition for multiple faces as depicted in the Figure 25.

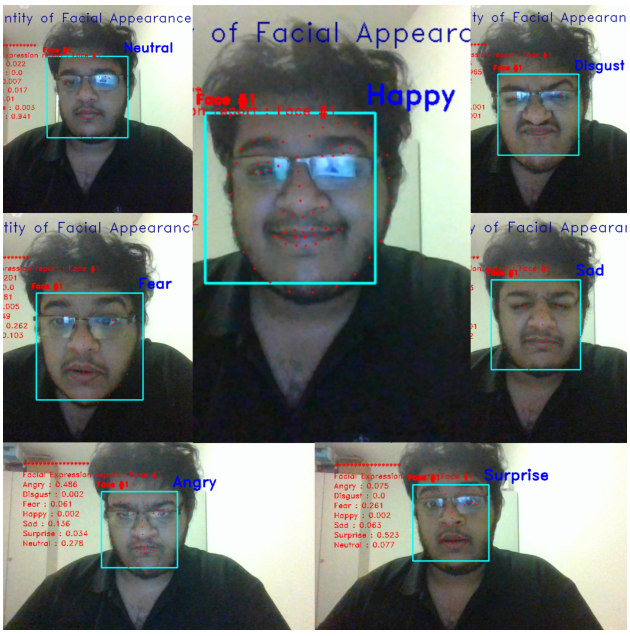


Figure 24: Seven Emotions Recognition illustration

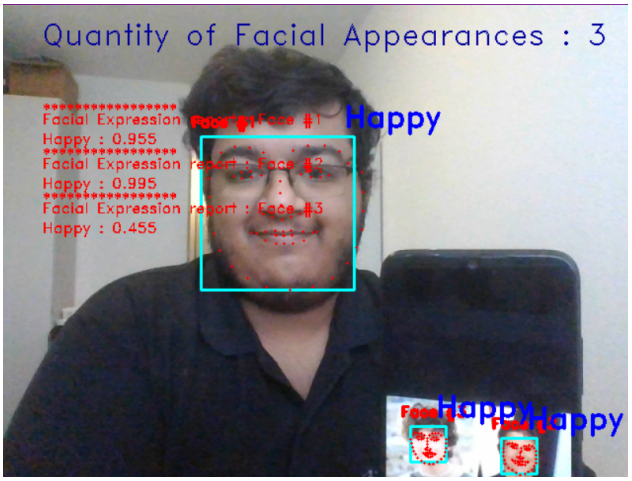


Figure 25: Multi Face Detection

The "SummarizeSingleModeEmotion" script which is part of code pattern serves the purpose of creating user session summaries by leveraging the information stored in the "predicted_visual_emotions.csv" file obtained from the earlier illustration session. This script is designed to offer valuable insights into the emotional dynamics of the user interactions. It achieves this by analyzing the predictions stored in the CSV file, particularly focusing on the occurrences of different emotions.

The primary outcome is a summary that sheds light on the emotions that were most frequently expressed by each user throughout the session. By scrutinizing the data, the script aims to highlight patterns and trends, revealing the predominant emotional states exhibited by users during their interactions. Additionally, the script goes beyond just identifying the most prevalent emotions; it also provides information about the least observed emotions during the sample session as depicted in the Figure 26. This comprehensive analysis contributes to a nuanced understanding of the emotional landscape within the user sessions, facilitating a more in-depth interpretation of user experiences and behaviors.

In the realm of future work, the implementation of context-encoded timestamping is poised to bring about heightened effectiveness. This innovative approach holds promise for capturing and understanding user behaviors within distinct contexts. Its potential lies in facilitating a more profound analysis of user emotional states, allowing for a comprehensive exploration of how individuals navigate and respond to specific situations. The integration of contextual information with timestamps is envisioned as a valuable avenue for delving deeper into user experiences, paving the way for enhanced insights into emotional dynamics. As part of the future work scope, this methodology holds the potential to significantly advance our understanding of user interactions and emotions.

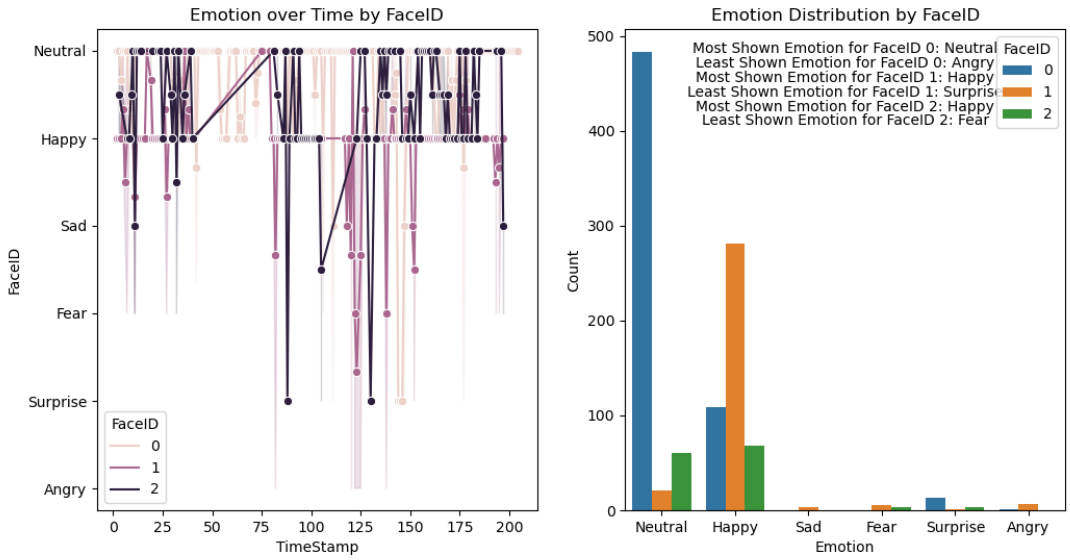


Figure 26: Visual Emotions Summary

5 Summary

In the pursuit of refining emotion recognition models, this project delved into the nuances of model architectures, dataset balancing, and feature extraction. The findings offer key insights and recommendations for future research.

Xception Models and Transfer Learning: Despite the computational demands of training Xception models, their substantial improvements over Inception underscore their potential in emotion recognition tasks. The adoption of transfer learning emerges as a pragmatic strategy, facilitating the adaptation of pre-trained models to specific tasks with increased efficiency and effectiveness.

Significance of Balanced Datasets: The project emphasizes the pivotal role of balanced datasets in fostering effective modeling. A balanced distribution ensures each emotion class receives equal representation, preventing biases and enhancing model generalization.

Challenges with Class Imbalance and Remedies: Imbalanced class distribution poses challenges, leading to biased model performance. Implemented strategies, such as oversampling, successfully address class imbalance, resulting in improved accuracy, precision, recall, and F1-score.

Stability in Training and Generalization: The incorporation of balanced datasets significantly contributes to a more stable training process. This ensures quicker convergence and mitigates overfitting, enabling models to develop a nuanced understanding of the entire dataset. Models trained on balanced datasets demonstrate enhanced generalization, fostering more reliable and applicable predictions.

Data Preprocessing Significance: The project underscores the critical role of data preprocessing in achieving successful outcomes in machine learning. Proper preprocessing enhances the quality and relevance of input data, influencing model performance.

Feature Evaluation and FER2013 Exploration: Feature extraction techniques, such as PCA and t-SNE, shed light on the data distribution in FER2013. Caution is advised against overly aggressive data reduction approaches to preserve meaningful patterns. Evaluation of different features, including HoG, facial landmarks, and combinations, highlights the importance of selecting features tailored to the specific characteristics of emotion recognition tasks.

Streamlined Annotation Process and Strategic Labeling: The implemented methodology for Yolo Processing of FER-2013 the Dynamic Annotation, which prioritizes annotations for images with successful face detection, stands out as a key preprocessing step. This selective approach enhances efficiency by circumventing explicit preprocessing for images lacking detected faces. Moreover, the introduced labeling strategy, involving the resolution of superset problems before annotating subsets, offers a systematic and resource-efficient solution. By addressing broader categories first and leveraging these solutions for specific subsets, the labeling process is not only optimized but also ensures accuracy and effectiveness in handling diverse datasets. This dual-pronged approach contributes significantly to the overall improvement of annotation workflow in image recognition tasks.

YoloV8 for Emotion Recognition: YoloV8 implementation showcases promising results for emotion recognition across diverse classes. Evaluation metrics, including precision, recall, mAP50, and mAP50-95, provide a comprehensive assessment of model performance. Future implementations could explore fine-tuning YoloV8 on emotion-specific datasets, potentially enhancing its ability to capture subtle facial expressions and nuances.

Comparative Analysis and Future Directions: Comparative analysis across various models and datasets reaffirms the substantial impact of dataset balancing on accuracy. Balanced datasets consistently outperform unbalanced counterparts, underscoring the importance of addressing class distribution. Future research could explore hybrid approaches, combining the strengths of different models, or delve into ensemble techniques to further improve emotion recognition accuracy and robustness. Additionally, ongoing efforts in refining YoloV8 for nuanced emotion detection present exciting avenues for exploration.

In summary, this project contributes valuable insights and recommendations for advancing emotion recognition models, addressing challenges, and suggesting future research directions for continued innovation in the field.

References

[1] F. Abdat, C. Maaoui, and A. Pruski. Human-computer interaction using emotion recognition from facial expression. In *Proceedings - UKSim 5th European Modelling Symposium on Computer Modelling and Simulation, EMS 2011*, 2011.

[2] P. Aiswarya, Manish, and P. Mangalraj. Emotion recognition by inclusion of age and gender parameters with a novel hierarchical approach using deep learning. In *2020 Advanced Communication Technologies and Signal Processing (ACTS)*, 2020.

[3] L. Alzubaidi, J. Zhang, A. J. Humaidi, and et al. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8(53), 2021.

[4] M. S. Bartlett, G. Littlewort, I. Fasel, and J. R. Movellan. Real time face detection and facial expression recognition: Development and applications to human computer interaction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 5, 2003.

[5] Marian Stewart Bartlett, Gwen Littlewort, Mark Frank, Cristina Lainscsek, Ian Fasel, and Javier Movellan. Fully automatic facial action recognition in spontaneous behavior. *Proceedings of the seventh IEEE international conference on automatic face and gesture recognition*, 2006.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153–160, 2007.

[7] S. Birogul, G. Temur, and U. Kose. Yolo object recognition algorithm and "buy-sell decision" model over 2d candlestick charts. *IEEE Access*, 8:91894–91915, 2020.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. 2020.

[9] M. Burić, M. Pobar, and M. Ivašić-Kos. Adapting yolo network for ball and player detection. In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, 2019.

[10] J. Chen, C. Wang, K. Wang, C. Yin, C. Zhao, T. Xu, X. Zhang, Z. Huang, M. Liu, and T. Yang. Heu emotion: A large-scale database for multimodal emotion recognition in the wild. *Neural Computing and Applications*, 33(14):8669–8685, 2021.

[11] François Chollet. Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[12] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893, 2005.

[14] Anatoli de Bradk, Mael Fabien, Raphael Lederman, and Stephane Reynal. Real-time multimodal emotion recognition. `https://github.com/maelfabien/Multimodal-Emotion-Recognition`, 2018-2019.

[15] R. Deepa, E. Tamilselvan, E. S. Abrar, and S. Sampath. Comparison of yolo, ssd, faster rcnn for real time tennis ball tracking for action decision networks. In *2019 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, 2019.

[16] B. Fasel and J. Luettin. Automatic facial expression analysis: A survey. *Pattern Recognition*, 36(1), 2003.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, et al. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[19] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[20] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] Ian T. Jolliffe. Principal component analysis. In *International Encyclopedia of Statistical Science*, pages 1094–1096. Springer, 2016.

[22] Davis E King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[24] Zhao Li, Shaogang Gong, and Caitlin Liddell. Facial expression recognition using gabor features and classifier fusion. *Pattern Recognition*, 46(1):279–288, 2013.

[25] G.-C. Luh, H.-B. Wu, Y.-T. Yong, Y.-J. Lai, and Y.-H. Chen. Facial expression based emotion recognition employing yolov3 deep neural networks. In *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, 2019.

[26] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579–2605), 2008.

[27] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[28] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. 2018.

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[30] S. Shinde, A. Kothari, and V. Gupta. Yolo based human action recognition and localization. *Procedia Computer Science*, 133:831–838, 2018.

[31] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, pages 1096–1103, 2008.