

Data Science Mini Project:

Emotion Recognition by Analysis of Facial Expressions in Datasets Acquired in Natural Environments (« In-The-Wild »)

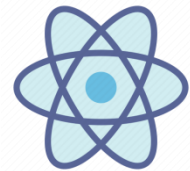
Introduction:

Facial Expression Recognition (FER) plays a crucial role in understanding human emotions, with applications ranging from human-computer interaction to affective computing. In this project, we aim to develop a robust image classification model to recognize seven emotions (joy, surprise, fear, anger, disgust, neutral and sadness) using the **FER 2013** dataset, which captures expressions in natural environments, referred to as "in-the-wild."

To do that we need to use different models such as : CNN , Resnet , GoogleNet, Inception and Vgg.

In this report we are going to present our work with these models and discover the results.

working environment:



Google Colab	Python	Visual Studio	React Js
<u>Google Colab for model development</u> served as the primary platform for developing and training the facial expression recognition models. Leveraging the power of cloud computing, Colab provided a fast and efficient environment with free access to GPU resources. The collaborative nature of	<u>Python for Model Development and Backend</u> a versatile and widely-used programming language, played a central role in both model development and backend implementation. The rich ecosystem of libraries, such as TensorFlow and	<u>Visual Studio for Backend and Frontend Development</u> provided a robust Integrated Development Environment (IDE) for backend development. Its extensive feature set, including powerful debugging tools and integrated source control, enhanced the	<u>React JS for Frontend Development</u> For crafting engaging and responsive user interfaces, React JS was employed for frontend development. React's component-based architecture and virtual DOM manipulation offered a modular and efficient approach to building interactive web applications. The use of

Colab allowed seamless collaboration with peers and facilitated version control using Google Drive.	PyTorch, enabled the implementation of complex neural network architectures for facial expression recognition. Additionally, Python's simplicity and readability streamlined the development of the backend application that consumes the trained models.	efficiency of writing server-side code.	React Bootstrap further expedited the creation of visually appealing frontend components.
---	---	---	---

Convolutional Neural Network (CNN):

A-Model Definition

The Convolutional Neural Network (CNN) developed for facial expression recognition is designed with multiple convolutional and fully connected layers. The model architecture is as follows:

```
# Config
input_shape = (48, 48, 1)
output_class = 7

# Model Definition
model = Sequential()

# Convolutional Layers
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

model.add(Conv2D(512, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

model.add(Conv2D(512, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

# Flatten Layer
model.add(Flatten())

# Fully Connected Layers
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))

# Output Layer
model.add(Dense(output_class, activation='softmax'))

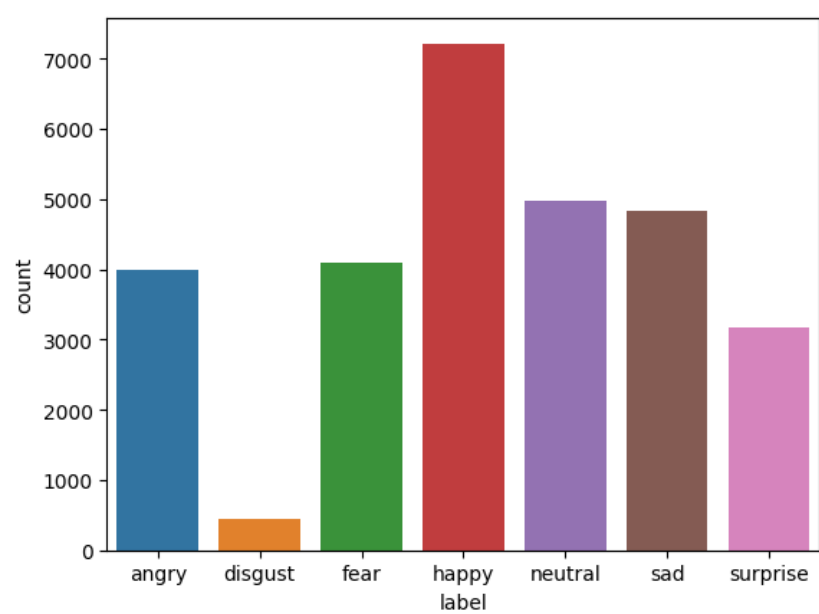
# Model Compilation
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics='accuracy')
```

The architecture includes:

- **Conv2D** (128 filters, kernel size 3x3, ReLU activation): Extracts low-level features.
- **MaxPooling2D** (2x2): Reduces spatial dimensions.
- **Dropout** (40%): Mitigates overfitting.
- **Conv2D** (256 filters, kernel size 3x3, ReLU activation): Extracts higher-level features.
- **MaxPooling2D** (2x2): Further reduces spatial dimensions.
- **Dropout** (40%): Reduces overfitting.
- **Conv2D** (512 filters, kernel size 3x3, ReLU activation): Enhances feature extraction.
- **MaxPooling2D** (2x2): Further reduces spatial dimensions.
- **Dropout** (40%): Addresses overfitting.
- **Conv2D** (512 filters, kernel size 3x3, ReLU activation): Deepens feature extraction.
- **MaxPooling2D** (2x2): Further reduces spatial dimensions.
- **Dropout** (40%): Controls overfitting.
- **Flatten**: Converts 2D feature maps to a 1D vector.
- **Dense** (512 neurons, ReLU activation): First fully connected layer.
- **Dropout** (40%): Mitigates overfitting.
- **Dense** (256 neurons, ReLU activation): Second fully connected layer.
- **Dropout** (30%): Controls overfitting.
- **Dense** (Output layer, 7 neurons, Softmax activation): Predicts emotion classes.

B- Data Exploration and Preprocessing

The dataset, FER 2013, consists of 7178 images illustrating seven different emotions. The dataset was loaded, and a random sample of images was visualized to gain insights into the distribution of emotions.



Images were preprocessed through grayscale conversion and normalized by dividing pixel values by 255. Labels were converted to integers and one-hot encoded using the LabelEncoder and to_categorical from scikit-learn and Keras, respectively.

C- Model Training

The CNN was trained using the preprocessed images and corresponding labels. The training process spanned 100 epochs, with a batch size of 128. The model's performance was evaluated on the validation set during training.

D-Monitoring Overfitting

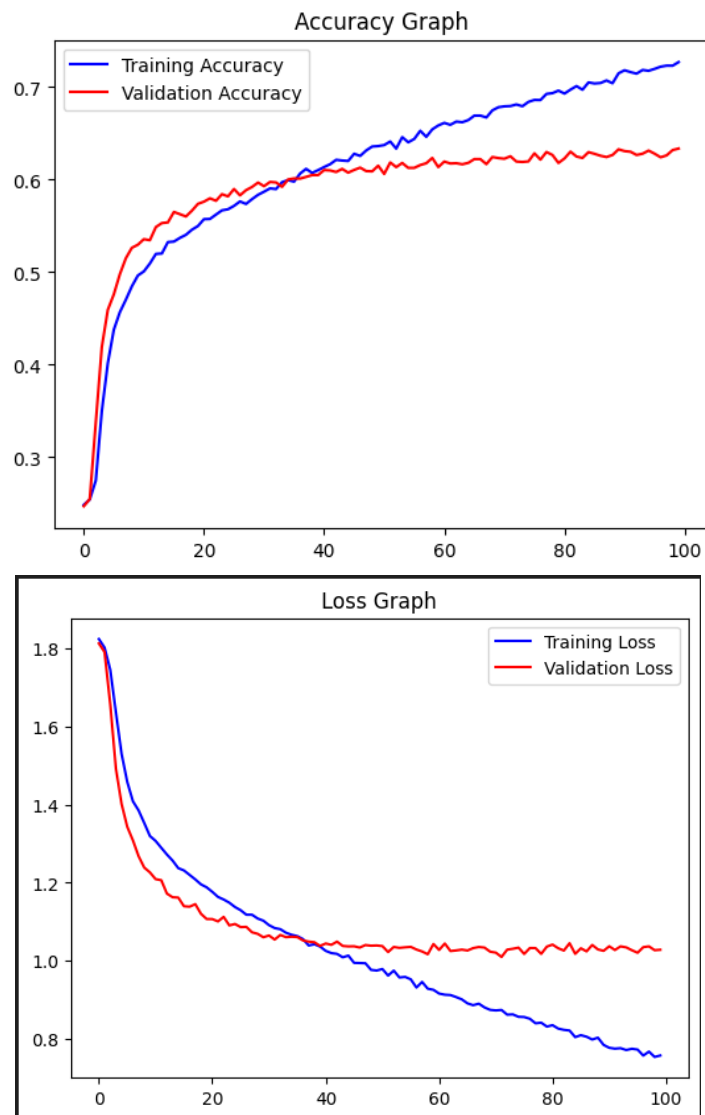
To monitor overfitting, both training and validation accuracy and loss were tracked over epochs. Dropout layers were strategically placed to reduce overfitting, especially in the fully connected layers.

E- Model Evaluation

The trained model was evaluated on the test set, yielding an accuracy of approximately **63.36%**. Evaluation metrics, including accuracy, loss, and confusion matrices, were used to assess the model's performance comprehensively.

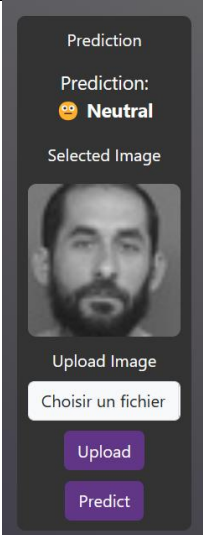

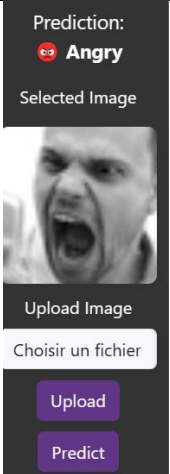
F- Model Visualization

Graphs displaying the training and validation accuracy and loss over epochs were generated for visual representation.



G-Sample Predictions

Random images from the test set were chosen to demonstrate the model's predictions. The model correctly predicted emotions such as "Neutral," "Fear," and "Angry," as shown in the sample predictions.

Original Output: Neutral, Predicted Output: Neutral	Original Output: Fear, Predicted Output: Fear	Original Output: Angry, Predicted Output: Angry
		

The trained PyTorch model was saved for integration in a front-end interface.

ResNet Model :

A-Model Definition

The ResNet model designed for facial expression recognition utilizes the ResNet34 architecture pretrained on ImageNet with a final fully connected layer adjusted for seven emotion classes. The key components include:

- Convolutional Base

1- **Convolutional Layer** (Conv2d):

Input: 3 channels (RGB)

Output: 64 filters, kernel size of (7, 7), stride of (2, 2)

Activation: Rectified Linear Unit (ReLU)

Function: Initial feature extraction, with a large kernel for capturing global features.

2- **Batch Normalization** (BatchNorm2d):

Normalizes the activations of the convolutional layers, improving stability and accelerating training.

3- **ReLU Activation** (ReLU):

Applies the rectified linear unit activation function, introducing non-linearity.

4- **MaxPooling Layer** (MaxPool2d):

Reduces spatial dimensions by performing max pooling with a kernel size of (3, 3) and stride of (2, 2).

- Fully Connected Layers:

The global average-pooled features are connected to fully connected layers for classification:

1- Fully Connected Layer 1:

Input features: Result of global average pooling

Output features: 7 classes (emotions)

Activation: Softmax

- Training:

The model is trained using cross-entropy loss, and the Adam optimizer is employed for weight updates during training. The learning rate is set to 0.001.

- Data Augmentation :

For data augmentation during training, random horizontal flips and rotations are applied.

- Evaluation Metrics

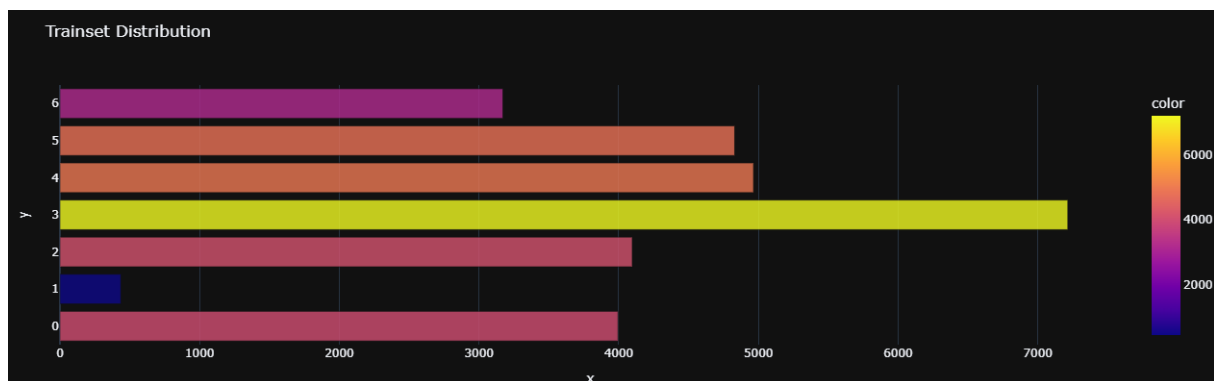
The model's performance is evaluated using metrics such as accuracy, loss, and confusion matrices.

- Sample Predictions

Random images from the validation set are used to showcase the model's predictions.

B- Data Exploration and Preprocessing

Similar to the CNN approach, the FER 2013 dataset was loaded, and data exploration was performed. Images were preprocessed using data augmentation techniques like random horizontal flips and rotations for the training set. Grayscale conversion and normalization were applied to both the training and validation sets.



C- Model Training

The ResNet model was trained over 20 epochs using Adam optimizer with a learning rate of 0.001 and a batch size of 32. The training process involved optimizing the model's weights based on the computed loss.




D-Monitoring Overfitting

To address overfitting, the training and validation accuracy and loss were monitored throughout the training process. Dropout layers were not required in the ResNet architecture due to the inherent residual connections, which help mitigate overfitting.

E- Model Evaluation

The trained ResNet model achieved a validation accuracy of approximately **64.97%**. Evaluation metrics, including accuracy and loss, were used to comprehensively assess the model's performance.

F- Sample Predictions

Original Output: Neutral, Predicted Output: Neutral	Original Output: Happy, Predicted Output: Happy	Original Output: Fear, Predicted Output: Fear
<div><div>Prediction</div><div>Prediction: 😐 Neutral</div><div>Selected Image</div><div></div><div>Upload Image</div><div>Choisir un fichier</div><div>Upload</div><div>Predict</div></div>	<div><div>Prediction</div><div>Prediction: 😊 Happy</div><div>Selected Image</div><div></div><div>Upload Image</div><div>Choisir un fichier</div><div>Upload</div><div>Predict</div></div>	<div><div>Prediction</div><div>Prediction: 😨 Fear</div><div>Selected Image</div><div></div><div>Upload Image</div><div>Choisir un fichier</div><div>Upload</div><div>Predict</div></div>

GoogleNet Model:

A-Model Definition

The GoogleNet model designed for facial expression recognition is built using the pre-trained GoogleNet architecture on ImageNet. The classification layer of GoogleNet is modified to match the specified number of classes (7 emotions in this case). The key components include:

- Convolutional Layers:
 - 1- **Conv2d (7x7, 64, stride=2, padding=3)**: 64 filters, 7x7 kernel size, stride of 2, and padding of 3.
 - 2- **MaxPool2d (3x3, stride=2)**: Max pooling with a 3x3 window and stride of 2.
- Inception Modules (Repeated Multiple Times):
 - 1- **Inception1 (64, 128, 128, 32, 32, 32)**: Multiple parallel paths with different filter sizes.
 - 2- **Inception2 (128, 192, 96, 32, 32, 64)**: Another set of parallel paths.
 - 3- **Inception3 (192, 208, 48, 32, 32, 64)**: Yet another set of parallel paths.

Note: Each path in the Inception modules may consist of 1x1 convolutions, 3x3 convolutions, and 5x5 convolutions.

- Auxiliary Classifiers (Added for Regularization):
 - 1- **Auxiliary Classifier 1**: A fully connected layer with dropout, predicting intermediate class scores.
 - 2- **Auxiliary Classifier 2**: Similar structure as Auxiliary Classifier 1.

Global Average Pooling:

- 3- **AvgPool2d (7x7)**: Global average pooling layer.
- Fully Connected Layers:
 - 1- **Linear (num_classes)**: Fully connected layer with the number of neurons equal to the number of output classes.

RQ: The original GoogleNet model is designed for ImageNet classification with 1000 classes. In this adaptation, the final fully connected layer is modified to output 7 classes corresponding to facial expressions.

B- Data Exploration and Preprocessing

The FER 2013 dataset was used for training and validation. Data augmentation techniques, such as random horizontal flips and rotations, were applied to the training set. Grayscale conversion and normalization were performed on both the training and validation sets.

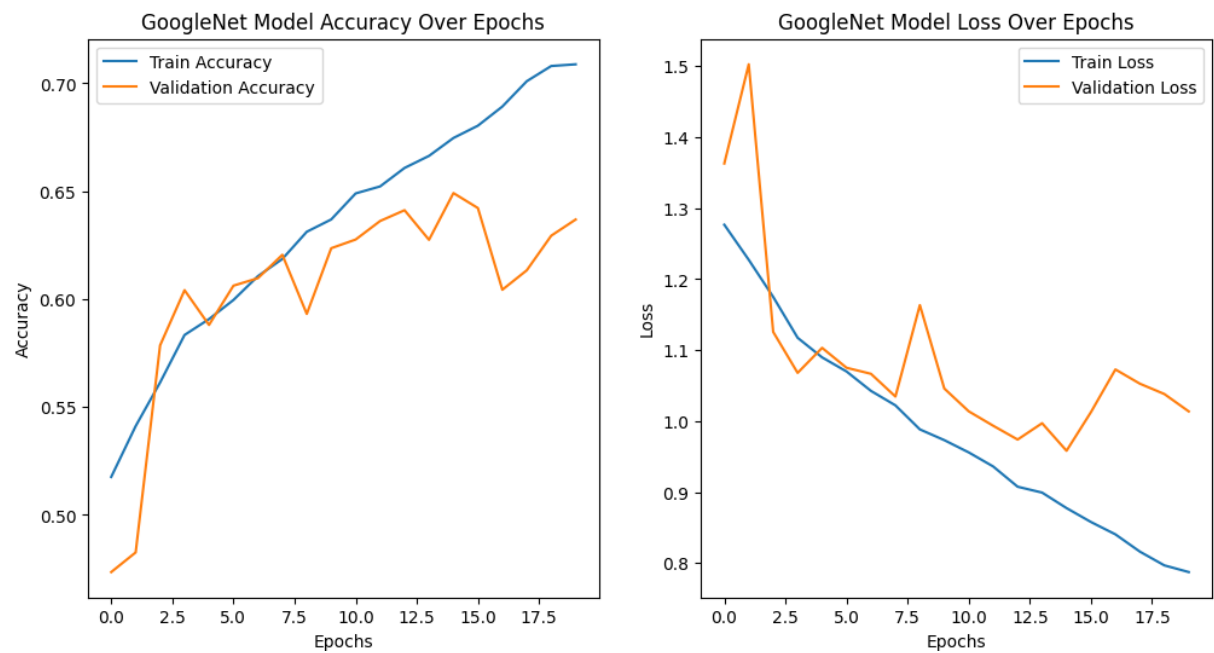
C- Model Training

The GoogleNet model was trained over 20 epochs using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The training process involved optimizing the model's weights based on the computed loss.

D- Monitoring Overfitting

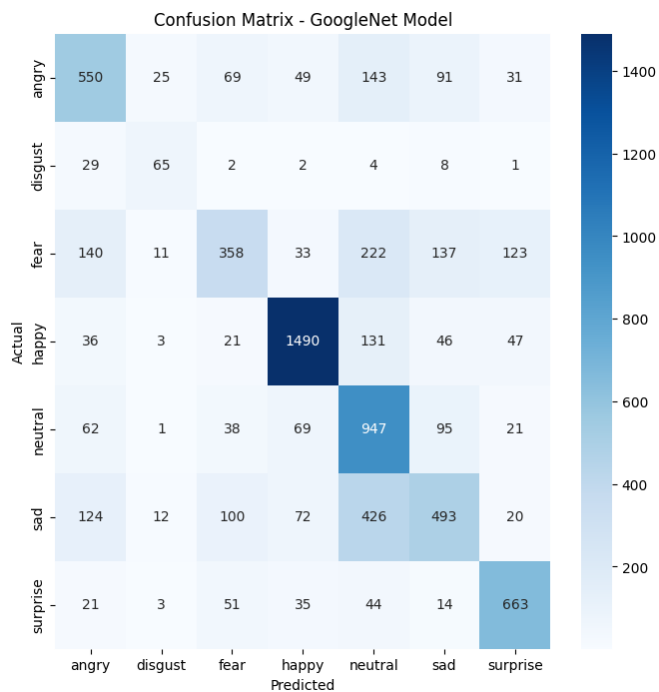
The training and validation accuracy and loss were monitored throughout the training process. No dropout layers were added, as GoogleNet typically includes inherent

regularization mechanisms. Graphs displaying the training and validation accuracy and loss over epochs were generated for visual representation.



E- Confusion Matrix


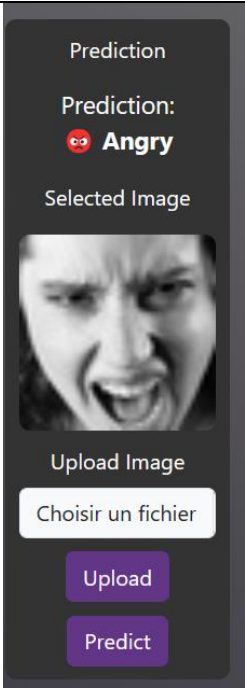
A confusion matrix was computed and visualized to show the distribution of predicted classes against actual classes.



F- Overall Accuracy

The overall accuracy of the GoogleNet model on the validation set was approximately **63.61%**.

G-Sample Predictions

Original Output: Happy, Predicted Output: Happy	Original Output: Angry, Predicted Output: Angry
	

Inception Model:

A-Model Definition

The Inception model for facial expression recognition is constructed using the pre-trained Inception V3 architecture. The Inception V3 model is adapted for the specific task of classifying facial expressions into seven categories. The last layer of the original Inception V3 model is replaced with a new fully connected layer with the appropriate number of output classes (7 emotions in this case). The key components include:

1- Input Layer:

Shape: Variable, depending on the input image size.

Description: The input layer receives the raw pixel values of the input image.

2- Preprocessing Layer:

Description: Preprocessing steps such as normalization and resizing may be applied to the input image.

3- Convolutional Layers (Basic Building Blocks):

Description: The core building blocks of the Inception model are the convolutional layers. These layers apply filters to the input image to extract hierarchical features.

4- Inception Modules:

Description: Inception modules are the key innovation in the Inception architecture. These modules use multiple parallel convolutional layers with different filter sizes to capture features at various scales.

5- Fully Connected Layer (Linear Layer):

Description: The output from the convolutional layers is flattened and connected to a fully connected layer. This layer transforms the extracted features into a format suitable for classification.

6- Output Layer (Softmax Activation):

Number of Neurons: 7 (for 7 emotion classes)

Activation Function: Softmax

Description: The final layer produces the probability distribution over the classes using the softmax activation function.

7- Auxiliary Classifiers (Optional):

Description: Inception V3 includes auxiliary classifiers at intermediate stages during training to aid in gradient flow and regularization. These classifiers have their loss functions.

8- Batch Normalization Layers:

Description: Batch normalization layers normalize the inputs of a layer, improving the training stability and accelerating convergence.

9- Pooling Layers (Average and Max Pooling):

Description: Pooling layers downsample the spatial dimensions, reducing the computational load and enhancing translation invariance.

10- Dropout (Optional):

Description: Dropout layers may be included for regularization purposes, helping prevent overfitting.

11- Global Average Pooling:

Description: Global average pooling reduces each feature map to a single value, providing a compact representation.

B- Data Exploration and Preprocessing

The FER 2013 dataset is utilized for both training and validation. Data augmentation techniques, such as random horizontal flips and rotations, are applied to the training set.

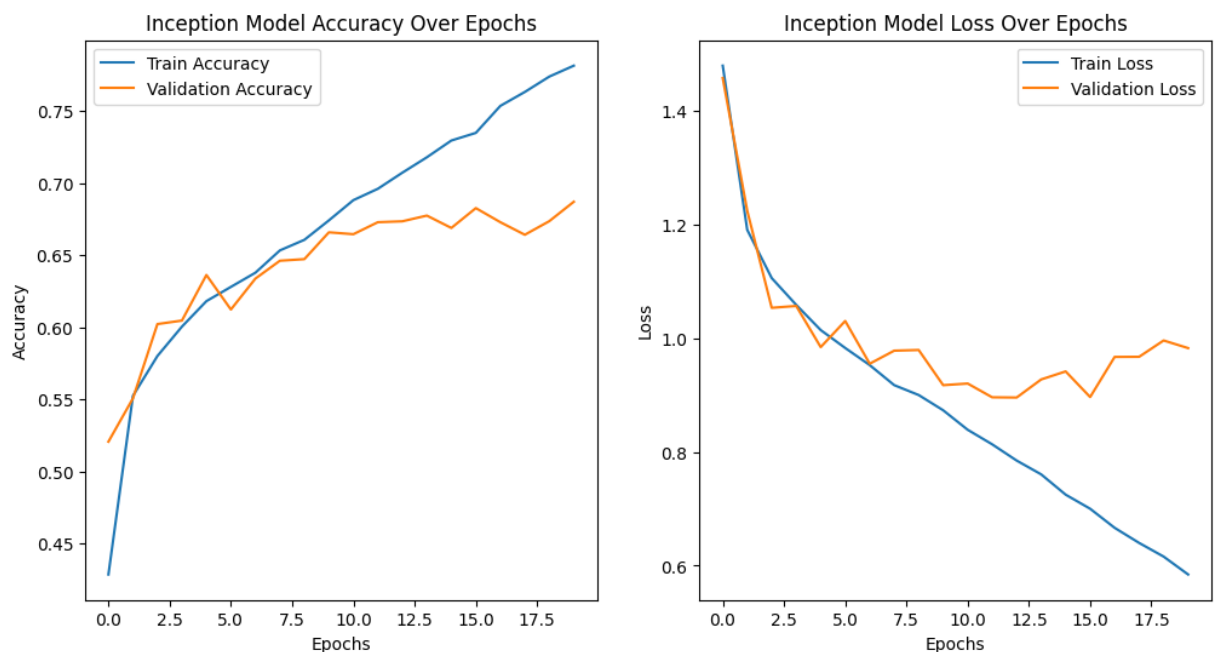
The dataset is loaded using PyTorch's ImageFolder class, and class distribution is visualized to ensure balanced representation.

C- Model Training

The Inception model is trained over 20 epochs using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The training process involves minimizing the cross-entropy loss.

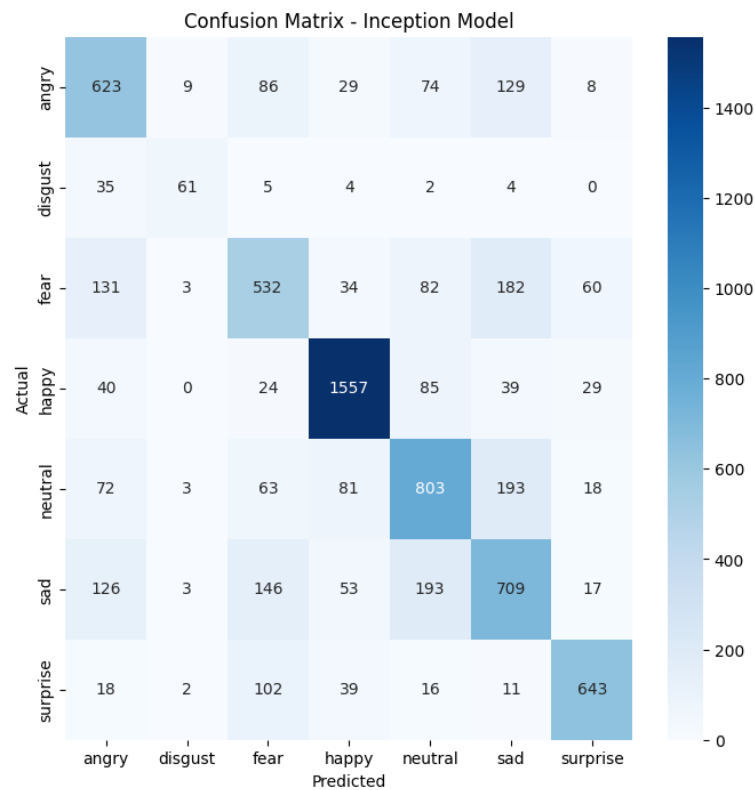
D- Monitoring Overfitting

The training and validation accuracy and loss are monitored during the training process. No explicit dropout layers are added, as Inception V3 includes its own regularization mechanisms. Graphs displaying the training and validation accuracy and loss over epochs are generated for visual representation.



E- Confusion Matrix




A confusion matrix was computed and visualized to show the distribution of predicted classes against actual classes.



F- Overall Accuracy

The overall accuracy of the GoogleNet model on the validation set was approximately **68.65%**.

H-Sample Predictions

Original Predicted	Output: Sad	Original Predicted	Output: Neutral	Original Predicted	Output: Angry
	Prediction Prediction: 😞 Sad Selected Image  Upload Image Choisir un fichier Upload Predict		Prediction Prediction: 😐 Neutral Selected Image  Upload Image Choisir un fichier Upload Predict		Prediction Prediction: 😡 Angry Selected Image  Upload Image Choisir un fichier Upload Predict

Vgg Model:

A-Model Definition

The VGG model for facial expression recognition is constructed using the pre-trained VGG16 architecture. The VGG16 model is adapted for the specific task of classifying facial expressions into seven categories. The last layer of the original VGG16 model is replaced with a new fully connected layer with the appropriate number of output classes (7 emotions in this case). The key components include:

- Input Layer:
 - 1- **The input to VGG16** is an RGB image of shape (224, 224, 3), where 3 represents the three color channels (red, green, and blue).
- Block 1:
 - 1- **Convolutional Layer** with 64 filters, each of size (3, 3), with a rectified linear unit (ReLU) activation function.
 - 2- **Convolutional Layer** with 64 filters, each of size (3, 3), with ReLU activation.
 - 3- **MaxPooling Layer** with a pool size of (2, 2) and a stride of (2, 2).
- Block 2:
 - 1- **Convolutional Layer** with 128 filters, each of size (3, 3), with ReLU activation.
 - 2- **Convolutional Layer** with 128 filters, each of size (3, 3), with ReLU activation.
 - 3- **MaxPooling Layer** with a pool size of (2, 2) and a stride of (2, 2).
- Block 3:
 - 1- **Convolutional Layer** with 256 filters, each of size (3, 3), with ReLU activation.
 - 2- **Convolutional Layer** with 256 filters, each of size (3, 3), with ReLU activation.
 - 3- **Convolutional Layer** with 256 filters, each of size (3, 3), with ReLU activation.
 - 4- **MaxPooling Layer** with a pool size of (2, 2) and a stride of (2, 2).
- Block 4:
 - 1- **Convolutional Layer** with 512 filters, each of size (3, 3), with ReLU activation.
 - 2- **Convolutional Layer** with 512 filters, each of size (3, 3), with ReLU activation.
 - 3- **Convolutional Layer** with 512 filters, each of size (3, 3), with ReLU activation.
 - 4- **MaxPooling Layer** with a pool size of (2, 2) and a stride of (2, 2).
- Block 5:
 - 1- **Convolutional Layer** with 512 filters, each of size (3, 3), with ReLU activation.
 - 2- **Convolutional Layer** with 512 filters, each of size (3, 3), with ReLU activation.
 - 3- **Convolutional Layer** with 512 filters, each of size (3, 3), with ReLU activation.
 - 4- **MaxPooling Layer** with a pool size of (2, 2) and a stride of (2, 2).
- Flatten Layer:

The output from the last max-pooling layer is flattened into a one-dimensional vector.

- Fully Connected Layers:
 - 1- Fully Connected Layer with 4096 units and ReLU activation.
 - 2- Fully Connected Layer with 4096 units and ReLU activation.
 - 3- Fully Connected Layer with 1000 units (output classes for ImageNet) and a softmax activation.

B- Data Exploration and Preprocessing

The FER 2013 dataset is utilized for both training and validation. Data augmentation techniques, such as random horizontal flips and rotations, are applied to the training set.

The dataset is loaded using PyTorch's ImageFolder class, and class distribution is visualized to ensure balanced representation.

C- Model Training

The VGG model is trained over 20 epochs using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The training process involves minimizing the cross-entropy loss.

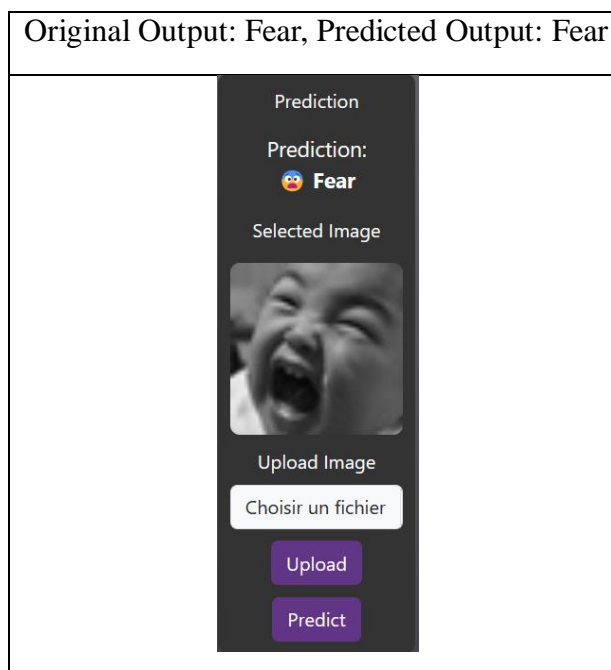
D- Monitoring Overfitting

The training and validation accuracy and loss are monitored during the training process. No explicit dropout layers are added, as VGG16 includes its own regularization mechanisms.

E- Overall Accuracy

The trained VGG model achieves a validation accuracy of approximately **24.64%**.

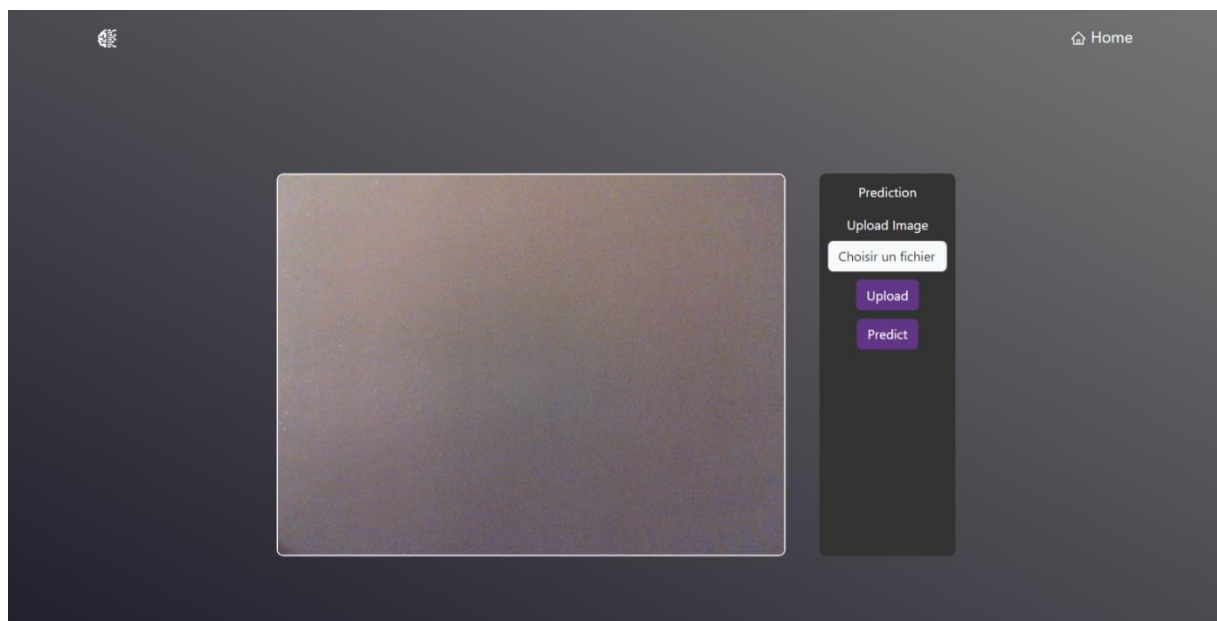
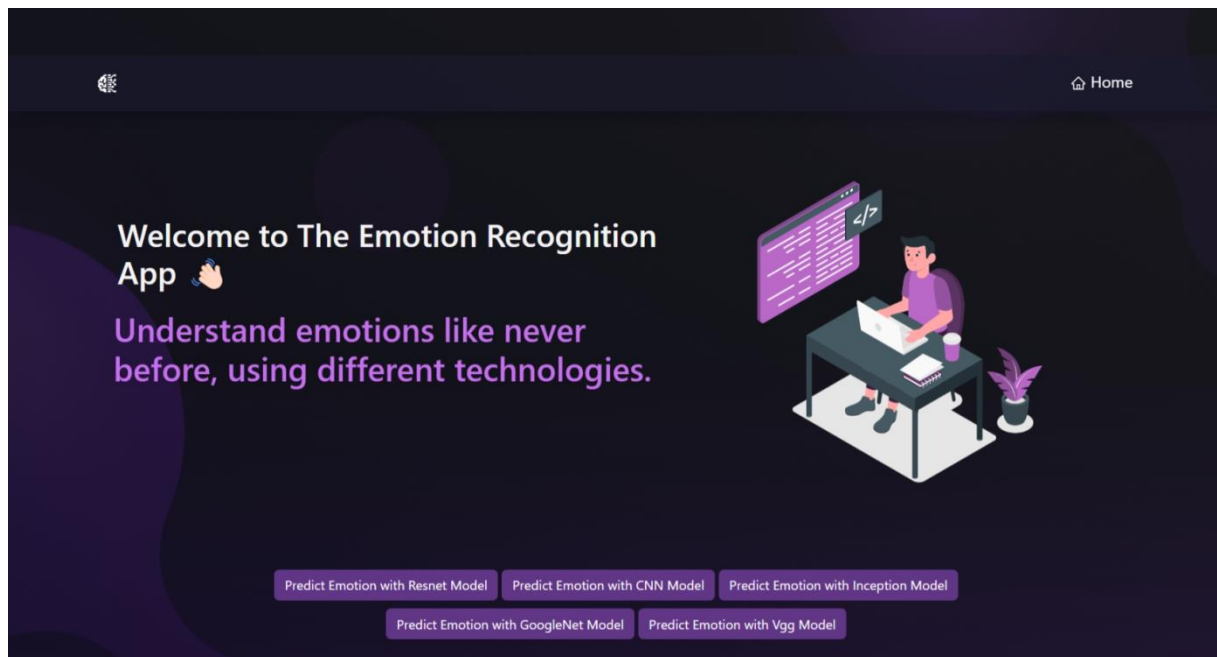
F- Sample Predictions



Graphical User Interface :

We developed a user-friendly graphical interface using React, coupled with a Python backend to enhance the interactive experience of utilizing the facial expression recognition model. The Python backend efficiently consumes the trained model and exposes its functionalities through a well-structured API. This API facilitates seamless communication with the React.js components, allowing for dynamic and real-time integration of the facial expression recognition capabilities into the user interface. The combination of React.js and Python not only ensures a responsive and intuitive user interface but also leverages the power of the pre-trained model for accurate and efficient facial expression analysis. This integration marks a

synergy between front-end and back-end technologies, delivering a cohesive and engaging user experience.



Conclusion:

In conclusion, this data science mini-project focused on developing and evaluating facial expression recognition models for in-the-wild datasets, specifically using the FER 2013 dataset. Five different models, namely CNN, ResNet, GoogleNet, Inception, and VGG, were explored and analyzed. Each model exhibited unique architectures, training processes, and performance metrics.

The Convolutional Neural Network (CNN) showcased a multi-layered architecture, emphasizing feature extraction and dropout layers to mitigate overfitting. The model achieved

a test accuracy of approximately 63.36%, showcasing its ability to recognize diverse facial expressions.

The ResNet model, built on residual connections, demonstrated robustness against overfitting. With a validation accuracy of around 64.97%, it highlighted the effectiveness of deep residual networks in facial expression recognition tasks.

GoogleNet, known for its inception modules, exhibited a validation accuracy of approximately 63.61%. The model's diverse convolutional paths showcased its adaptability to capture intricate facial features.

Inception, utilizing the Inception V3 architecture, achieved the highest validation accuracy among the models, reaching approximately 68.65%. The incorporation of auxiliary classifiers and inception modules contributed to its superior performance.

On the other hand, the VGG model, with its straightforward architecture, showed a validation accuracy of approximately 24.64%. Despite its simplicity, VGG struggled to capture nuanced facial expressions compared to other models.

While each model demonstrated unique strengths, the Inception model's combination of architectural innovation, regularization mechanisms, and robust training process led to its outstanding performance in this specific facial expression recognition task.

Finally, a graphical user interface (GUI) was developed using React.js and a Python backend to provide an interactive platform for utilizing the trained models. This seamless integration of front-end and back-end technologies ensured a user-friendly experience for facial expression analysis.