**1. Introduction**

This project presents a system designed for detecting laughter in real-time using a custom version of the **LeNet** deep learning architecture. The system is part of an interactive game where players try not to laugh while being shown funny media. The model processes live webcam video feeds to detect laughter based on facial expressions, enabling a real-time gameplay experience.

**2. Objective**

The objective of this project is to develop a real-time **laugh detection game** where players attempt not to laugh. Using a trained deep learning model, the system monitors players' facial expressions and determines when they are laughing. The game then tracks the duration of laughter and determines a winner based on who laughs the least.

**3. Dataset: GENKI-4K**

The model was trained on the **GENKI-4K** dataset, which contains **4000 images** of faces. This dataset is labeled with two types of information:

* **Expression labels**: Indicating whether the subject is smiling or not smiling.
* **Head-pose labels**: Providing data about the orientation of the head (pose).

The **GENKI-4K** dataset is particularly useful for smile detection because of its variety in expressions and head poses, making it an ideal choice for training a model to detect laughter in different facial orientations.

**4. LeNet Architecture Overview**

The deep learning model used for laugh detection is a custom adaptation of the **LeNet** architecture, specifically designed to process grayscale images of faces (64x64 pixels). The architecture incorporates modern enhancements such as **Leaky ReLU** activations, **batch normalization**, and **dropout** layers to improve performance and prevent overfitting.

Key architecture details:

* **Convolutional layers**:
  + First layer: Conv2D with 32 filters (5x5 kernel) → Leaky ReLU → MaxPooling.
  + Second layer: Conv2D with 64 filters (5x5 kernel) → Leaky ReLU → MaxPooling.
* **Fully connected layers**:
  + Flattening the feature maps → Dense (512 units) → Leaky ReLU → Dropout.
* **Output layer**:
  + Dense layer (2 units) → Softmax for binary classification (laughing or not laughing).

**5. Training the Model**

**5.1 Data Preprocessing**

The training process involved several key steps to prepare the dataset:

* **Face Detection**: **Haar Cascade** was used to detect and crop faces from the images in the dataset. These cropped face images were then resized to 64x64 pixels.
* **Normalization**: The pixel values of the images were normalized to the range [0, 1] to improve training efficiency.

**5.2 Data Augmentation**

To increase the model's robustness and ability to generalize, data augmentation techniques were applied, including:

* **Random rotations** (up to 30 degrees),
* **Width and height shifts** (up to 10%),
* **Shearing** (up to 15%),
* **Zooming** (up to 25%),
* **Horizontal flipping**.

These augmentation techniques ensured that the model learned to recognize laughter across a variety of facial orientations and expressions.

**5.3 Class Imbalance Handling**

The GENKI-4K dataset is slightly imbalanced, with more non-smiling images than smiling ones. To address this, **class weighting** was applied during training, ensuring the model paid equal attention to both classes.

**5.4 Training Configuration**

* **Optimizer**: **Adam** optimizer with a learning rate of 0.001 was used.
* **Loss Function**: **Binary Cross-Entropy** was employed for the classification task.
* **Learning Rate Adjustment**: **ReduceLROnPlateau** was used to adjust the learning rate when validation performance plateaued.

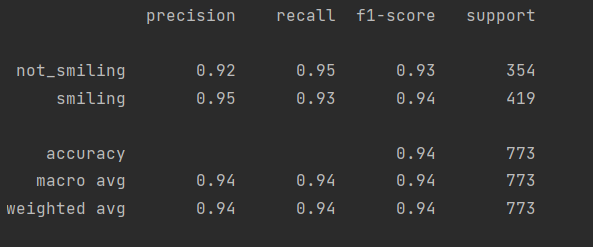
**Training Parameters:**

* **Epochs**: 70 epochs.
* **Batch Size**: 32.
* **Validation Split**: 20% of the dataset was reserved for validation during training.

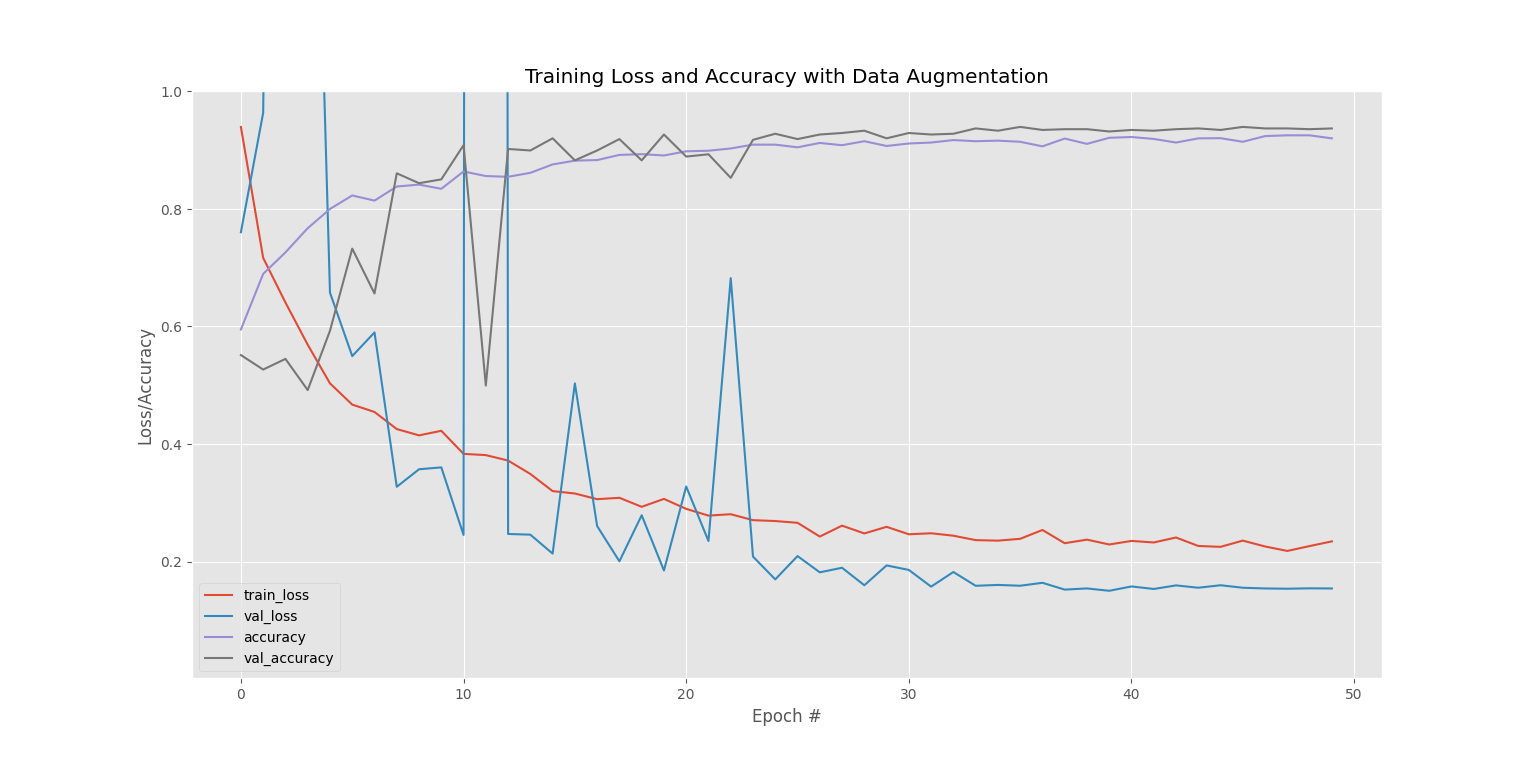
**6. Evaluation and Results**

The model was evaluated using the test set, and the following performance metrics were computed:

* **Precision, Recall, and F1-score**: For both the "laughing" and "not laughing" classes, to assess the model's ability to accurately classify the expressions.

****

**\*(not\_smiling = not\_laughing, smiling = laughing)**

* **Training History**: The loss and accuracy curves across 70 epochs were plotted to show the model's learning progress. ****

These graph illustrates how the model's performance evolved during training, providing insights into convergence and overfitting.

**7. User Interface (UI) Design**

The game interface was created using the **Tkinter** framework, offering a simple and intuitive UI that allows players to engage in the "Try Not to Laugh" challenge. The key features include:

* **Player Mode Selection**: The game supports both **1-player** and **2-player** modes.
* **Customizable Game Duration**: Players can set the length of the game (e.g., 30 or 60 seconds).
* **Funny Media**: Players can choose to include funny videos or images during the game to make it more challenging.

**Game Logic:**

* **1-Player Mode**: The system tracks the total time the player spends laughing, and displays the results at the end of the game.
* **2-Player Mode**: The game tracks the laughter duration of both players and declares the winner as the player who laughed the least.
* **Real-Time Updates**: The UI continuously updates the game status, displaying whether players are laughing, along with a timer showing how much time remains.

**8. Real-Time Laugh Detection Workflow**

1. **Face Detection**: The system uses a webcam to capture live video. **Haar Cascade** is employed to detect faces in each frame.
2. **Laugh Classification**: The detected faces are resized to 64x64 pixels and passed through the trained LeNet model to classify whether the player is laughing or not.
3. **Game Scoring**: The system tracks how long each player laughs and updates the score in real-time based on the classification results.

**9. Conclusion**

This project successfully integrates a custom deep learning model with a game interface to detect laughter in real-time. Using the **GENKI-4K** dataset for training, the model performs well in classifying facial expressions of laughter. The interactive game design adds a fun and engaging layer to the challenge of not laughing while being provoked by funny media.

**10. Future Work**

There are several ways to improve and extend this project:

* **Expanding the Dataset**: Additional datasets could be incorporated to further enhance the model’s generalizability to different facial expressions and ethnicities.
* **Game Enhancements**: A potential enhancement to the game would be adding and saving face embeddings for each player. This would allow the system to recognize previous players who may have left mid-game and resumed later without restarting their laugh counters. The face embeddings could be used to match players when they return, ensuring continuity in the game. However, this feature would require significant resources, as it would need to check the embeddings in each frame to identify the player. Implementing this efficiently would be challenging without the necessary GPU resources to handle the computational load in real-time.

This project showcases the power of machine learning for real-time applications, blending technology with entertainment to create an enjoyable and interactive experience.