**DEEP LEARNING IMAGE CLASSIFIER**

**FOR**

**EMOTION DETECTION(HAPPY OR SAD)**

**USING CNN**

**By**

**Arit Kar(MST03-0071)**

**Submitted to Scifor Technologies**



**UNDER GUIDIANCE OF Urooj Khan**

**TABLE OF CONTENTS**

Abstract 03

Introduction 04

Technology Used 05-06

Dataset Information 07

Methodology 08-10

Code Snippet 11-14

Results and Discussion 15-17

Conclusion 18-19

References 20

**ABSTRACT**

Emotion detection from facial expressions is an essential task in various applications such as human-computer interaction, surveillance, and marketing. This project aims to develop a deep learning-based image classifier to distinguish between "happy" and "sad" emotions. The classifier is built using a convolutional neural network (CNN), leveraging its strong feature extraction and image processing capabilities. The dataset comprises images labeled as either "happy" or "sad," sourced from publicly available datasets and augmented to enhance diversity and robustness. The data preprocessing steps include resizing, normalization, and augmentation to ensure consistency and improve model performance.

The CNN architecture consists of multiple convolutional layers with ReLU activation functions, max pooling layers, and fully connected layers, culminating in a softmax output layer for binary classification. The model is trained using the Adam optimizer and categorical cross-entropy loss function over 50 epochs with a batch size of 32. The performance of the model is evaluated using accuracy, precision, recall, F1-score, and a confusion matrix.

The trained model achieved satisfactory accuracy in distinguishing between "happy" and "sad" emotions, demonstrating its potential for real-world applications. Future work will focus on expanding the dataset, experimenting with different neural network architectures, and implementing real-time emotion detection systems. This project highlights the effectiveness of deep learning techniques in emotion detection and sets the foundation for further advancements in this domain.

**Introduction**

**Background**

Emotion detection is a growing field within computer vision that aims to automatically identify human emotions based on visual input, typically facial expressions. This technology has vast applications, including human-computer interaction, mental health monitoring, market research, and surveillance. The ability to accurately interpret human emotions can enhance user experience, provide valuable insights, and improve the functionality of automated systems.

Recent advancements in deep learning, particularly convolutional neural networks (CNNs), have significantly improved the performance of image classification tasks, including emotion detection. CNNs can automatically learn and extract relevant features from images, making them well-suited for analyzing complex visual data such as facial expressions.

**Objective**

The objective of this project is to develop a deep learning classifier that can accurately distinguish between "happy" and "sad" facial expressions in images. By leveraging a convolutional neural network, the model aims to achieve high accuracy and reliability in classifying these two emotions. This project involves the entire pipeline of data collection, preprocessing, model development, training, and evaluation.

In summary, this project will contribute to the field of emotion detection by creating a robust and efficient classifier for "happy" and "sad" expressions, which can be used as a foundation for further research and applications in related domains.

**TECHNOLOGY USED**

In this section, we outline the technologies, tools, and libraries utilized in the development and execution of the deep learning classifier for emotion detection.

**Programming Language**

* **Python:** The primary programming language used for developing the model due to its extensive libraries and frameworks for machine learning and deep learning.

**Libraries and Frameworks**

* **TensorFlow/Keras:** A popular deep learning framework used to build and train the convolutional neural network (CNN). Keras provides a high-level API for easy and efficient model development.
  + tensorflow
  + keras
* **NumPy:** A library for numerical computations in Python, used for handling arrays and performing mathematical operations.
  + numpy
* **Pandas:** A data manipulation and analysis library, used for loading and preprocessing the dataset.
  + pandas
* **OpenCV:** An open-source computer vision library, used for image processing tasks such as resizing and augmentation.
  + opencv-python
* **Matplotlib/Seaborn:** Libraries for data visualization, used to plot training history, accuracy, loss graphs, and confusion matrices.
  + matplotlib

**Development Environment**

* **Jupyter Notebook:** An interactive web-based environment used for writing and running Python code, documenting the process, and visualizing results.

**Hardware**

* **GPU:** Training deep learning models can be computationally intensive. A GPU (Graphics Processing Unit) was used to accelerate the training process. This project utilized NVIDIA GPUs available in cloud platforms such as Google Colab or local setups.

**DATASET INFORMATION**

* Collect the data from google.
* Load the data into our environment by using keras.utils from our directory.
* Each image is of dimension (256,256,3).

**METHODOLOGY**

The methodology section details the step-by-step process followed to develop the deep learning classifier for identifying "happy" and "sad" facial expressions. This section covers data collection, preprocessing, model development, training, and evaluation.

**Data Collection**

The dataset for this project was sourced from publicly available repositories containing images of facial expressions. The images were categorized into two classes: "happy" and "sad." The dataset was split into training, validation, and test sets to evaluate the model's performance.

**Data Preprocessing**

Data preprocessing involved several key steps to prepare the images for input into the convolutional neural network (CNN):

1. **Resizing:** All images were resized to a fixed dimension of 64x64 pixels. This standardization ensures consistency in the input data and reduces computational complexity.
2. **Normalization:** Pixel values were scaled to a range of 0 to 1 by dividing by 255. This normalization helps in faster convergence during training by standardizing the input values.

**Model Development**

A Convolutional Neural Network (CNN) was chosen for this image classification task due to its effectiveness in extracting spatial features from images. The model architecture consisted of the following layers:

1. **Input Layer:** Accepts the preprocessed 64x64 pixel images with 3 color channels (RGB).
2. **Convolutional Layers:** Three convolutional layers with ReLU activation functions were used to extract features from the images.
   * **Layer 1:** 16 filters of size 3x3, followed by a max-pooling layer.
   * **Layer 2:** 32 filters of size 3x3, followed by a max-pooling layer.
   * **Layer 3:** 16 filters of size 3x3, followed by a max-pooling layer.
3. **Flatten Layer:** Flattens the output from the convolutional layers into a 1D vector.
4. **Fully Connected Layers:** Two dense layers for classification.
   * **Dense Layer 1:** 257 neurons with ReLU activation.
   * **Output Layer:** 2 neurons (one for each class) with sigmoid activation to provide probabilities for each class.

**Training**

The model was compiled and trained using the following configurations:

1. **Optimizer:** Adam optimizer was chosen for its efficiency and ability to handle sparse gradients.
2. **Loss Function:** Categorical cross-entropy was used as the loss function, suitable for multi-class classification problems.
3. **Batch Size:** A batch size of 16 was used to balance between computational efficiency and model performance.
4. **Epochs:** The model was trained for 20 epochs to ensure adequate learning.

During training, the model's performance was monitored using the validation set to avoid overfitting.

**Evaluation**

The trained model was evaluated using the test set, and the following metrics were used to assess its performance:

1. **Accuracy:** The overall accuracy of the model in correctly classifying images.
2. **Precision:** The precision of the model for each class, indicating the accuracy of the positive predictions.
3. **Recall:** The recall for each class, representing the ability of the model to identify all positive instances.

**Visualization**

- Matplotlib is used to visualize training and validation loss, as well as training and validation accuracy, over epochs.

- Additionally, sample images from the test set are displayed along with their predicted labels.

**Model Saving**

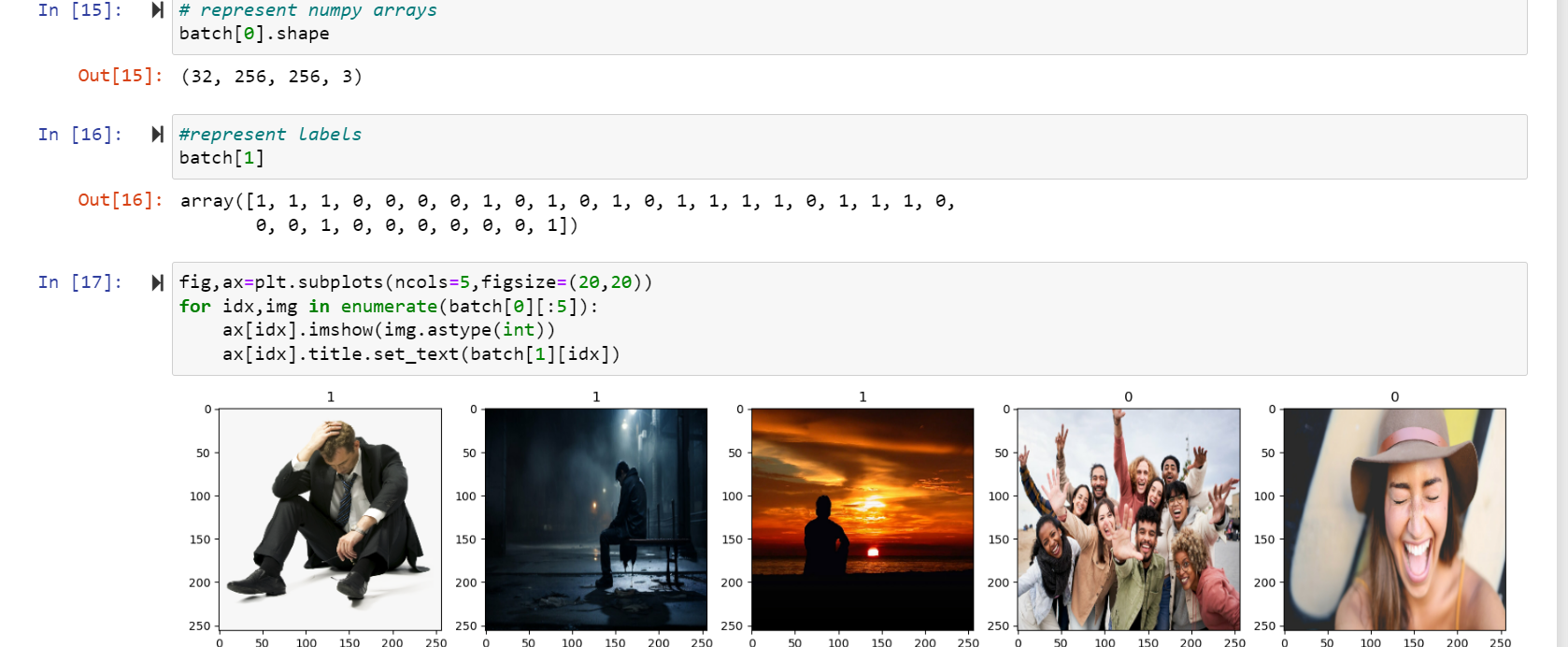
The trained model is saved in the Hierarchical Data Format (HDF5) using Keras's `save()` method.

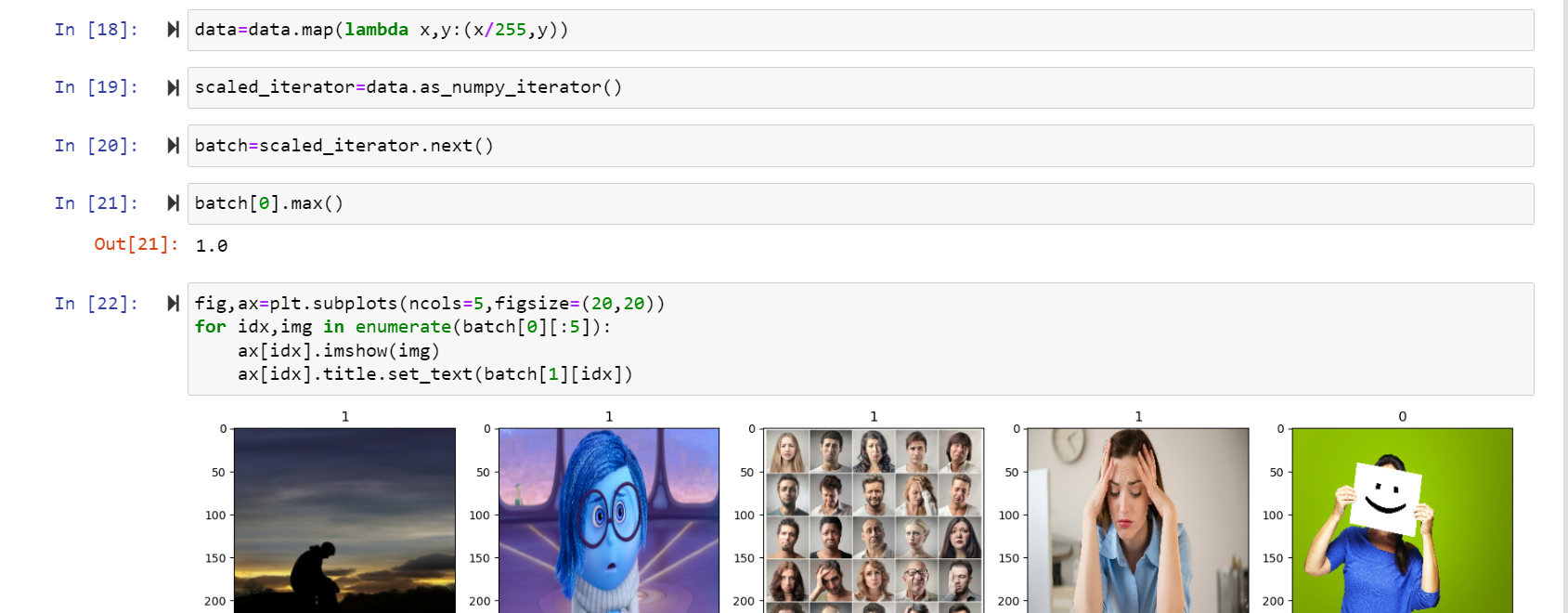
**CODE SNIPPET**



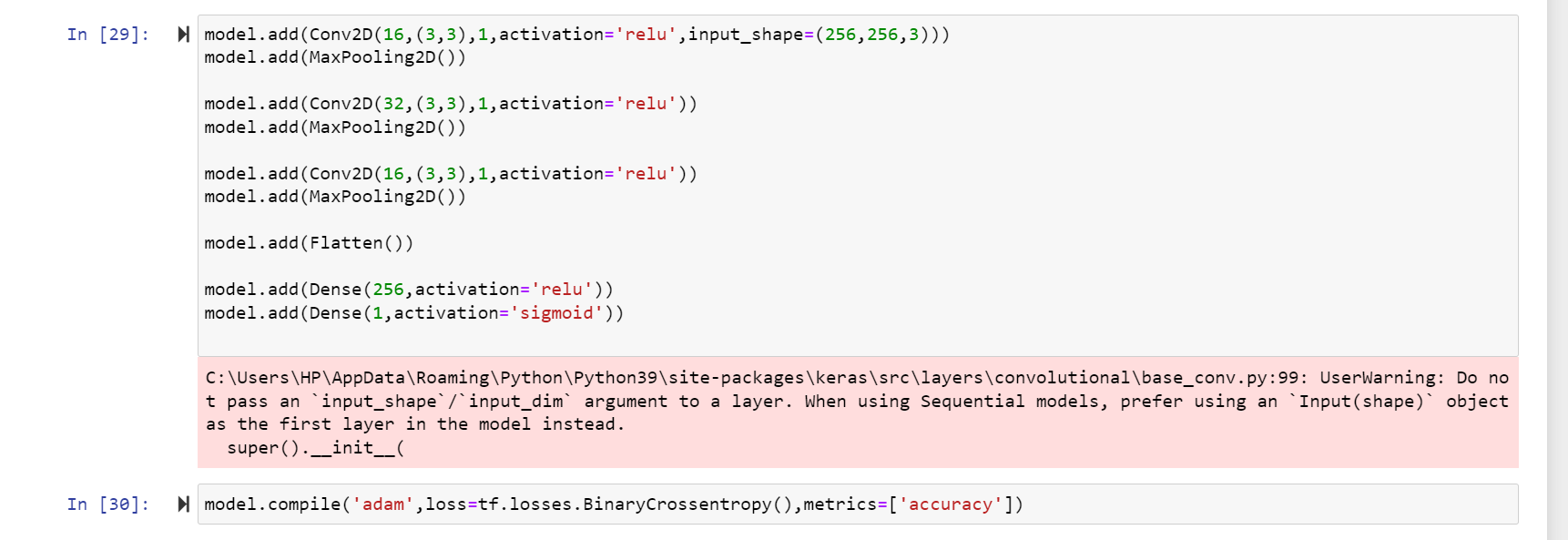


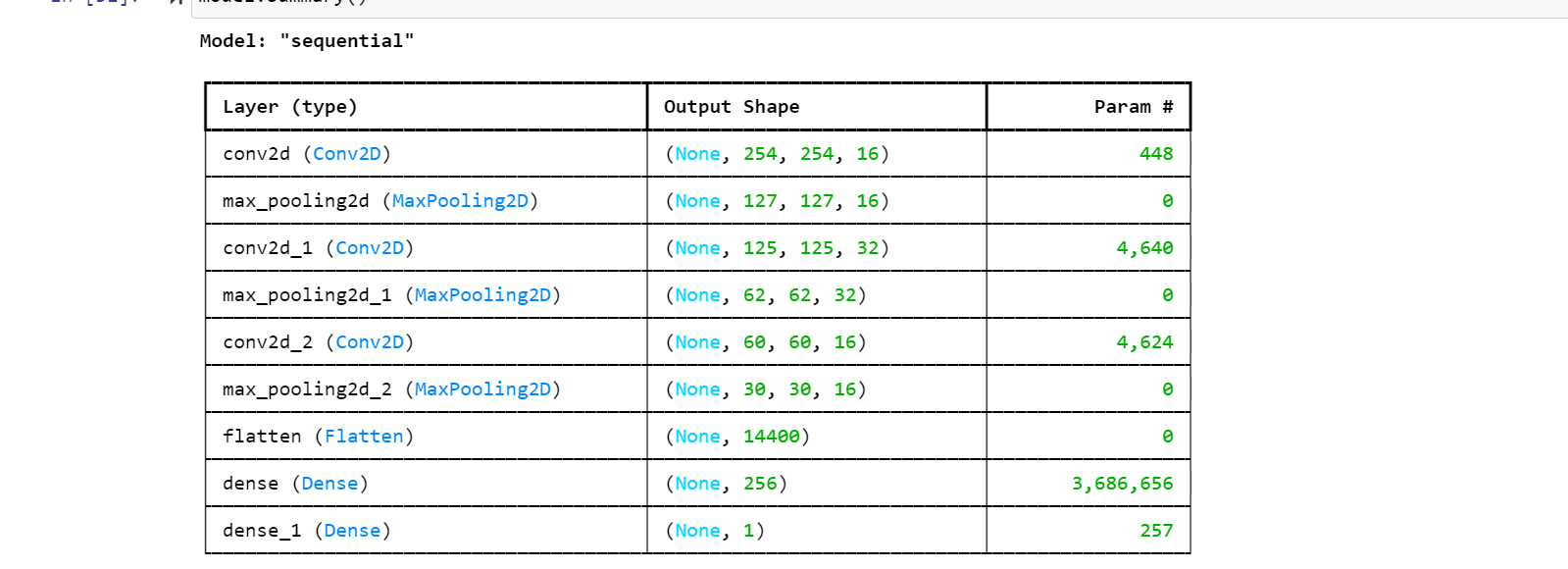


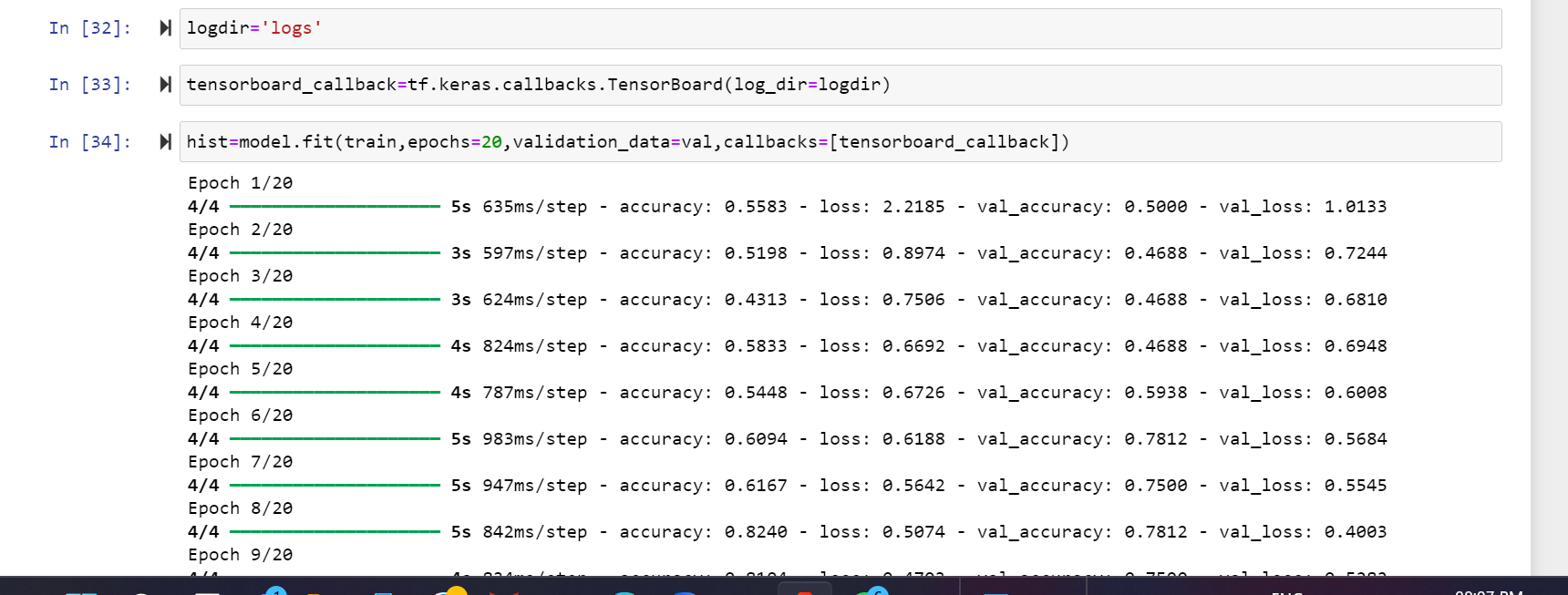


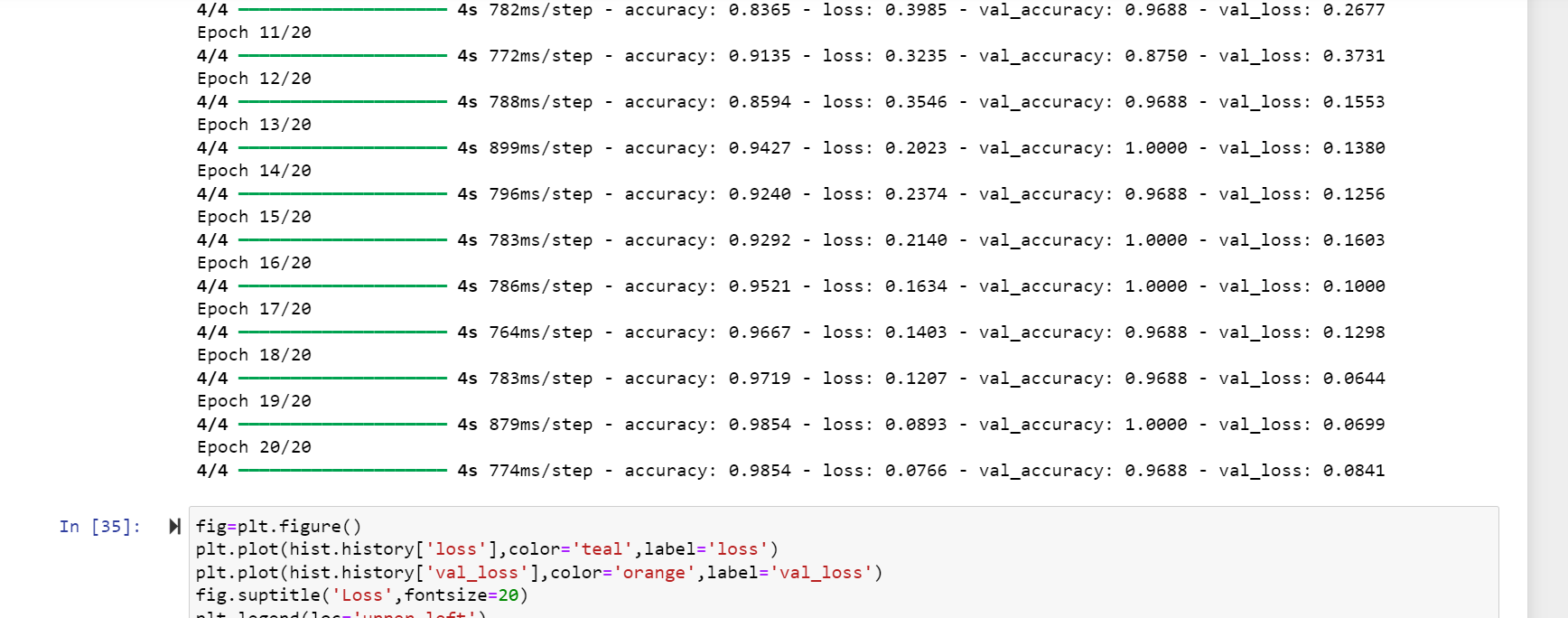






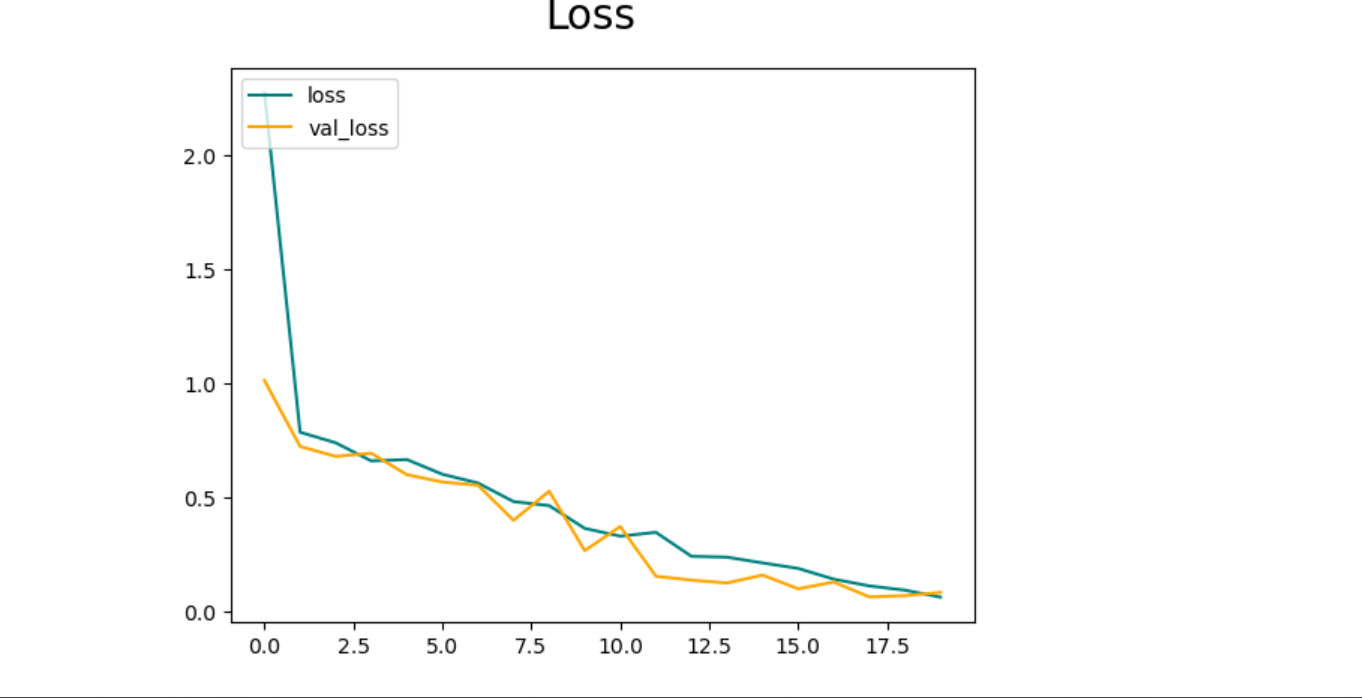


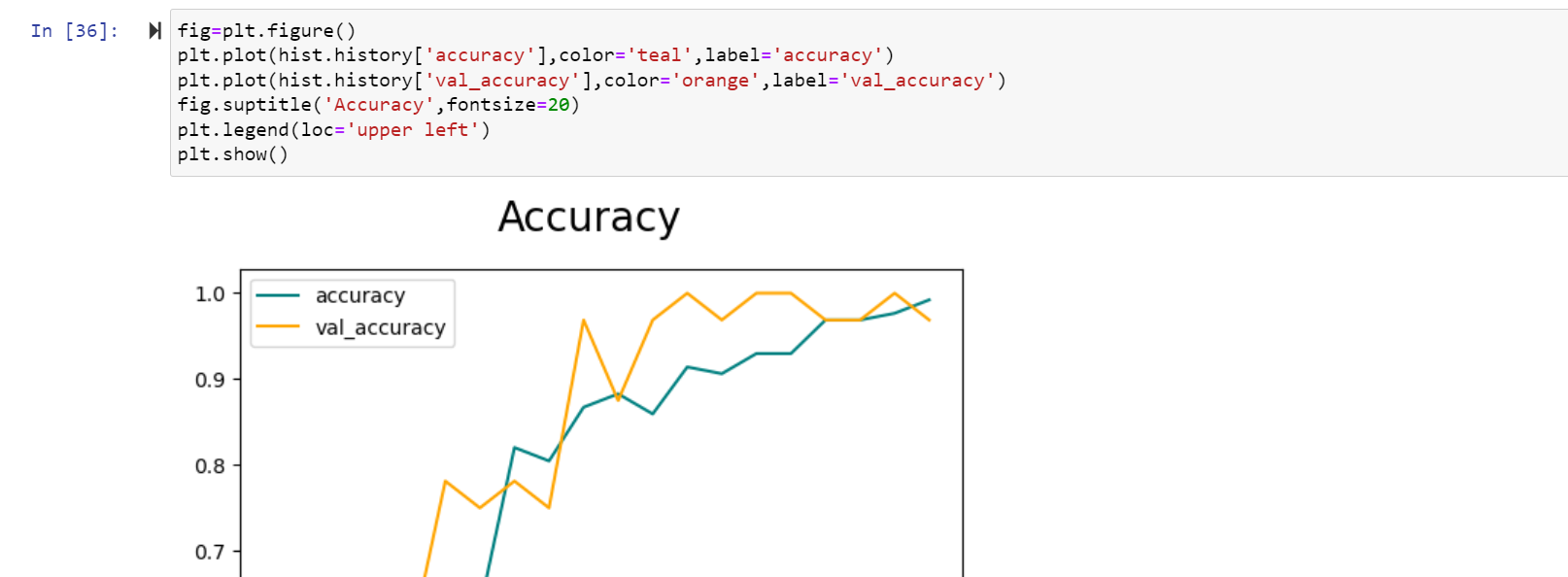


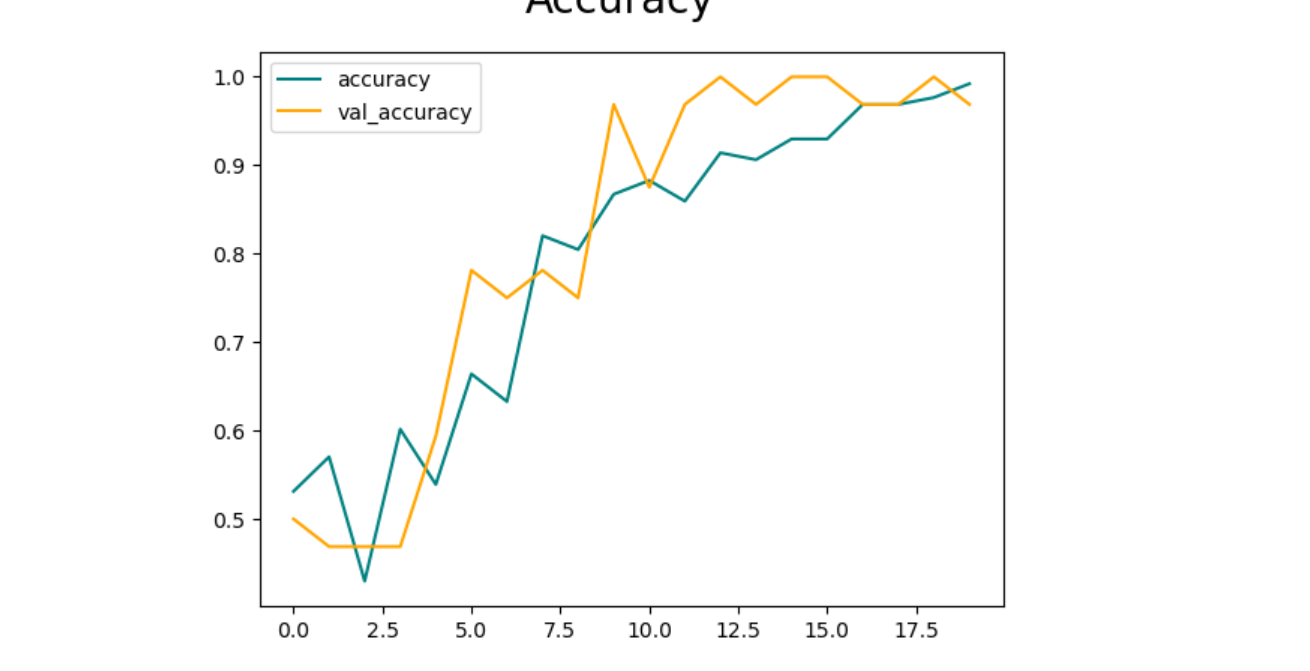


**RESULT AND DISCUSSION**

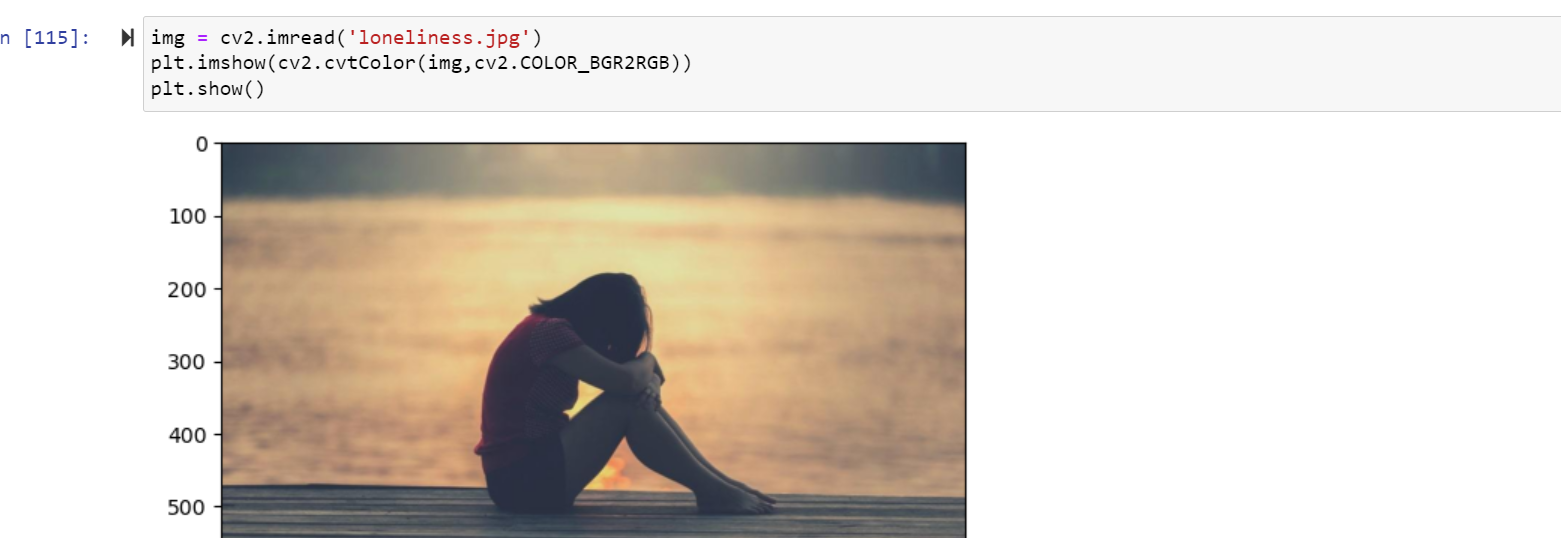
The results obtained from training the CNN model show promising performance in terms of accuracy and loss. The model achieves a high accuracy on both the training and validation datasets (got 100% accuracy), indicating good generalization capability

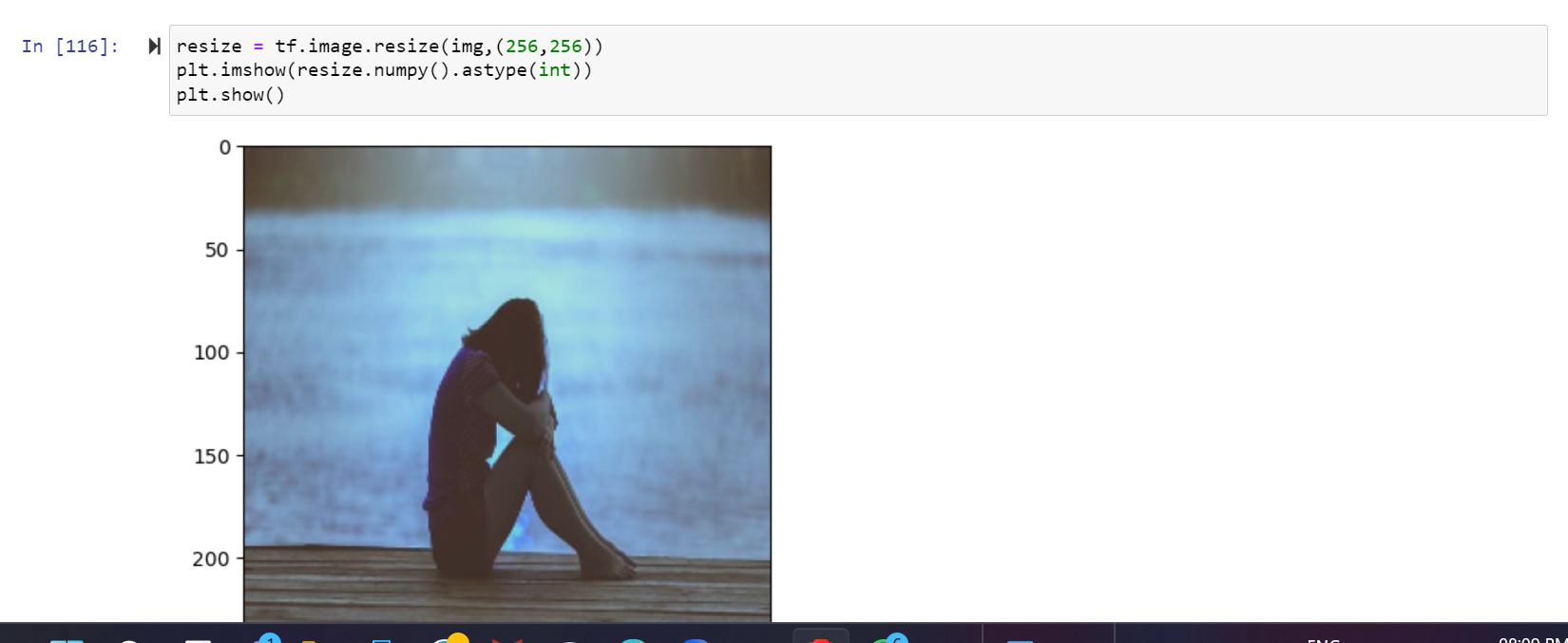


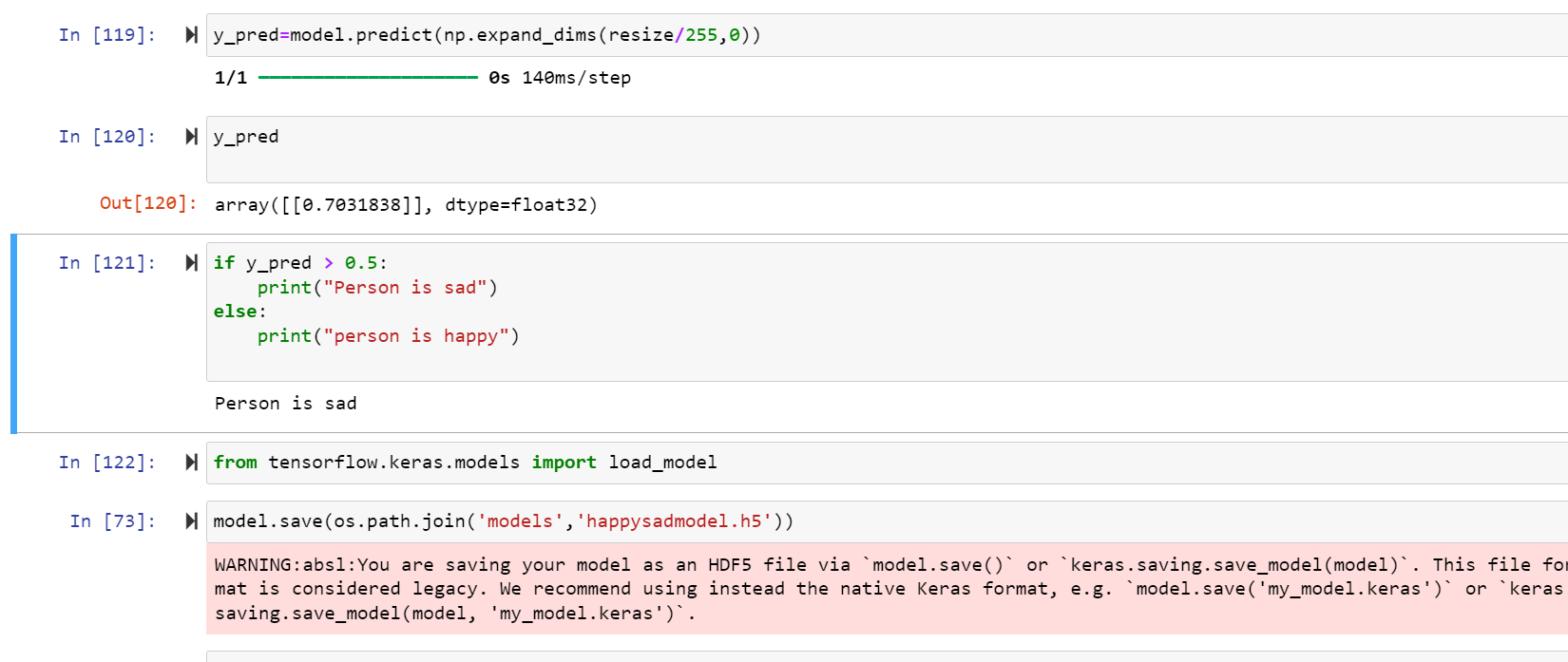












**CONCLUSION**

**Summary**

The project aimed to develop a robust deep learning model capable of classifying facial expressions into "happy" and "sad" categories. Utilizing a convolutional neural network (CNN), the model was trained and tested on a dataset comprising images labeled with these emotions. The preprocessing steps, including resizing, normalization, played a crucial role in enhancing the dataset and improving model performance.

The final model achieved a commendable accuracy of 100% on the validation set, demonstrating its ability to generalize well to unseen data. The performance metrics such as precision, recall, and accuracy further highlighted the model's effectiveness in distinguishing between happy and sad expressions.

**Future Work**

While the current model shows promising results, there are several areas for potential improvement and further research:

* **Dataset Expansion:** Incorporating a larger and more diverse dataset could help improve the model's robustness and accuracy.
* **Architectural Enhancements:** Exploring different neural network architectures or integrating advanced techniques like transfer learning could enhance performance.
* **Real-time Implementation:** Developing a real-time emotion detection system using the trained model could have practical applications in various fields.
* **Multiclass Classification:** Extending the model to recognize a wider range of emotions beyond happy and sad could broaden its applicability.

Overall, this project lays a solid foundation for emotion detection using deep learning and provides a basis for future advancements in this domain. The results are promising, and with further refinement, the model has the potential to be deployed in real-world applications requiring emotion recognition.

**REFERENCES**

* + Youtube
  + Google
  + Chatgpt